MINIORDENADOR

PARTE 1

PRESENTACION GENERAL PARA LOS NIVELES I Y II

DEL MICROORDENADOR TRS-80

Del mismo modo que la aplicación de la máquina trajo consigo la revolución industrial, la incorporación de la informática a nuestro mundo está suponiendo una transformación de incalculables dimensiones.

Nuestra época no pasará a la Historia como la que dió comienzo a la Era Atómica, sino más bien como aquella que permitió una de las mutaciones más profundas del medio social, gracias a la incorporación de la informática.

No es éste el lugar adecuado para hacer un análisis sociológico de una ciencia, que todavía apenas ha iniciado su andadura en nuestro país. Pero el hecho está ahí y es irreversible.

Para quienes nos dedicamos a la educación y al desarrollo científico-técnico, la informática es un reto, un tren en marcha que no podemos perder ni detener. Es preciso tomarlo si no queremos ser arrollados por él. Después podremos intentar conducirlo.

La mejor manera de introducirse en el conocimiento de la Informática es ir haciendo uno mismo sus propios programas, teniendo posibilidad de ejecutarlos y modificarlos a la vista de los resultados.

Este modo de trabajar posee características de indudable incidencia en el campo educativo, algunas de las cuales queremos resaltar a continuación.

En primer lugar, estimula la atención de modo natural, eliminando el mecanismo emisor-receptor, que a menudo, convierte al alumno en un ser pasivo y sin iniciativa.

El aprendizaje de la programación no supone una lucha contra la máquina, sino una superación continua del programador. La máquina es ciega, pero enormemente rápida. El programador va tomando conciencia paulatina de cómo esta rapidez, supone una herramienta eficaz para su propio aprendizaje. El estudiante cae en la cuenta de que crea aprendiendo y de que aprende creando. Esto produce una sensación de gozo que estimula nuevas búsquedas y nuevos aprendizajes.

En segundo lugar, un adecuado aprendizaje de la programación supone un desarrollo mental importante, así como, la creación de hábitos que permiten la ordenación metódica del pensamiento lógico.

Hace poco más de una década se estableció en Francia la obligatoriedad del estudio del Algebra de Boole y de sus aplicaciones inmediatas en todos los centros de enseñanza.

Así pues, no se perseguía sólo conocer una técnica aplicada, cosa que incumbiría a la enseñanza profesional, sino también y sobre todo, conseguir un desarrollo mental.

Esta es la razón por la que opinamos que el aprendizaje de la programación debe tener un lugar también en nuestro B.U.P.

En tercer lugar, el modo conversacional de operar con la máquina informática puede ser muy útil para la adquisición de conceptos en el campo matemático, físico y aún tecnológico, que, de otro modo, suponen un esfuerzo de abstracción en el vacío, que pocos escolares pueden realizar.

Naturalmente, esto supone una orientación didáctica importante por parte del profesor de la materia en cuestión, profesor que habrá de conocer, además, el lenguaje de programación utilizado.

Considerando que, de momento, la gran mayoría de profesores de una materia determinada no tiene conocimientos de programación, hemos incluido en nuestros manuales un importante capítulo destinado a la comprensión de algunos conceptos matemáticos y físicos.

Todas las características citadas hasta ahora están dirigidas en el TRS-80 NIVEL I, tanto para estudiantes de FP2 como de B.U.P.
La cuarta característica a considerar es la utilidad práctica de la informática.

El abanico de aplicaciones es inmenso, pero vamos a ceñirnos al campo científico-técnico en sus dos especialidades mecánica y eléctrica. En cada una de ellas existen técnicas en las que la aportación de la informática es fundamental. Tales son, el control de mando numérico y los microprocesadores.

El control numérico está transformando la máquina-herramienta convencional, a la que quizás llegue a desterrar.

Los microprocesadores permiten automatizar cualquier tipo de proceso industrial o de la vida ordinaria, siendo además el núcleo generador de la robótica y aún de la misma informática.

El TRS-80 NIVEL II posee la potencia suficiente para incidir en los campos citados y en otros, pudiendo citar entre éstos como ejemplo más relevante, el de la organización y gestión empresariales.

Este nivel sería el correspondiente a escuelas de Ingeniería, aunque no descartamos la posibilidad de que pueda ser utilizado en FP2 si el nivel previo se desplaza a etapas anteriores.

Una quinta característica de la informática es la posibilidad de almacenar programas, cuya ejecución puede ahorrar tareas rutinarias. Tal es el caso de la resolución de sistemas de ecuaciones algebraicas, sistemas de ecuaciones diferenciales, métodos de aproximación de curvas, desarrollo en serie de Fourier, etc...

Esta característica del ordenador o sistema informático como "regla de cálculo potente", cuya hermana mediana sería la calculadora de bolsillo programable, ha sido preponderante en las aplicaciones de la informática en el campo científico-técnico.

La influencia de esta característica en el campo estrictamente pedagógico o didáctico es mucho menor, requiriendo por otra parte, el concurso de equipos más sofisticados. También se cubre en proporción suficiente con el TRS-80 NIVEL II.

APARTADO 1º

INTRODUCCION

PROLOGO AL TRS-80 NIVEL I

Este manual ha sido escrito pensando en el estudiante que no sabe nada sobre computadoras y desea introducirse en el mundo de la programación de manera rápida, sencilla, eficaz y hasta divertida.

El sistema TRS-80 ofrece al usuario la posibilidad de programación en lenguaje BASIC; este tipo de lenguaje destaca, dentro del amplio espectro existente, por su sencillez unida a una aceptable potencia y flexibilidad. Estas cualidades le acreditan como uno de los más usados en las escuelas media y superior y de gran utilidad en aplicaciones empresariales, técnicas y científicas.

Se pretende que este manual vaya orientado hacia niveles de enseñanza tales como FP2 y B.U.P.; por esto nos hemos propuesto en su elaboración tres finalidades :

a) Desarrollo de la capacidad de razonamiento del escolar.

b) Ayuda en la comprensión de las distintas materias de enseñanza.

c) Posibilitar el uso del TRS-80 como herramienta de trabajo en el campo científico-técnico.

Finalidad a).- La filosofía de programación BASIC, al igual que cualquier otro lenguaje, ofrece al alumno un desafío a su imaginación y capacidad de razonamiento. Nosotros pretendemos que este desafío sea gradual, de manera que el lector vaya integrando conocimientos a medida que avanza en el aprendizaje de las nuevas instrucciones de programación.

Para llevar a cabo nuestro propósito, a lo largo de los capítulos del manual se van proponiendo y resolviendo ejercicios fáciles, que irán creciendo en dificultad a medida que se progresa en el conocimiento del lenguaje BASIC, para culminar con un apartado de ejercicios de juegos. Hemos elegido este apartado de juegos porque pensamos que es una buena forma de motivar al aprendizaje de la programación BASIC. Es importante que el lector vaya tecleando y ejecutando los programas para conseguir un aumento de la confianza en sí mismo y estímulo por el tema.

Finalidad b).- El alumno con conocimientos de programación BASIC, está en franca ventaja a la hora de asimilar determinadas materias de que se compone su curso académico. Este manual permite utilizar el microordenador TRS-80 como ayuda a la comprensión de esas partes más dificultosas de matemáticas, física, etc... que por su nivel de abstracción, resultan de más difícil asimilación.

Para conseguir esta finalidad incluimos un apartado dedicado a la resolución de problemas y puntos teóricos dificultosos en materias de enseñanza fundamentalmente numéricas. Una gran ventaja del ordenador estriba, en que permite una visualización secuencial en la ejecución de un proceso (en nuestro caso un problema matemático, por ejemplo); ideas como la de derivada o integral adquieren, mediante el tratamiento secuencial-incremental, una nueva perspectiva que colabora en la asimilación del concepto.

Finalidad c).- El iniciado en programación BASIC, es capaz ya de intuir las enormes posibilidades de su microordenador.

Trabajos de cálculo extremadamente dificultosos y largos de ejecutar, dejarán de serlo al ser editados y ejecutados en su sistema TRS-80, ya que una vez grabados en cinta, pasarán a formar parte de su biblioteca de programas y estarán listos para ser utilizados como herramienta de cálculo cuando sea preciso. Los apartados finales de este manual, contienen una variada gama de programas de utilidad práctica. Se sugiere el estudio de dichos programas para completar el aprendizaje de la programación BASIC NIVEL I.

PUESTA A PUNTO DEL SISTEMA

PRIMEROS CONSEJOS

Desempacar cuidadosamente el sistema. Quitar todo el material de empaque y guardarlo para cuando se necesite transportar el sistema. Asegurarse de localizar todos los cables, papeles, cintas, etc...

CONEXION DE LA PANTALLA DE VIDEO Y TECLADO

1. Conectar el cable de la pantalla de video a una toma de c.a. de 220 V. 50 ciclos.
2. Conectar el cable del alimentador a una toma de c.a. de 220 V. 50 ciclos.
3. Conectar el cable gris del monitor de video al borne "VIDEO", que está localizado en la parte trasera del teclado. Asegurarse de conectar bien los terminales (el conector sólo entra bien en un sólo sentido).

   NOTA :

   Antes de efectuar el siguiente paso, asegurarse de que el interruptor "POWER", que se encuentra en la parte trasera del teclado, esté apoyado (el botón hacia fuera).
4. Conectar el cable gris del alimentador al borne "POWER" que se encuentra en la parte trasera del teclado. Una vez más asegurarse de unir los conectores correctamente.

CONEXION DE LA GRABADORA

NOTA :

No es necesario conectar la grabadora de cinta a menos que se necesite grabar programas o cargar programas al TRS-80.

1. Poner las baterias al CTR-80 (grabadora) como se describe en el manual, ó efectuar la conexión a la línea c.a. de 220 V.

2. Conectar el cable corto (conector DIN en un extremo y 3 conectores en el otro extremo), al borne TAPE en la parte trasera del teclado.

   Asegurarse de efectuar la conexión correctamente.

3. Los 3 conectores del otro extremo de este cable, son para efectuar las conexiones al CTR-80.

   A. Enchufar el conector negro al borne EAR que se encuentra en un costado del CTR-80. Esta conexión, proporciona la señal de salida del CTR-80 al TRS-80, (para cargar los programas de la cinta al TRS-80).

   B. Enchufar el conector gris de clavija más gruesa, al borne AUX del CTR-80. Esta conexión proporciona la señal de grabación para grabar programas del TRS-80 al CTR-80.

   Además enchufar el conector falso (suministrado con el CTR-80) al borne "MIC" (esto desconecta el micrófono interno para que no recoja sonidos mientras se están cargando cintas).

   NOTA :

   Asegurarse de usar siempre el conector falso mientras se graban programas en cinta.

C. Enchufar el conector gris de clavija más delgada al borne REM en el CTR-80. Esto permite que el TRS-80 controle automáticamente el motor del CTR-80 (encender y apagar al movimiento de la cinta para grabar ó cargar los programas).

NOTAS ACERCA DEL USO DE LA GRABADORA

Existen varios puntos que se deben tener en cuenta cuando se utilize el sistema de cinta (más adelante, se dará sobre esto mayor información, cuando se hable del uso del casette de cinta).

1. Para utilizar la cinta (cargar un programa grabado al TRS-80), deberá colocarse el control del volumen de la CTR-80 en 7 u 8. Luego se presiona la tecla PLAY del CTR-80 y se teclea CLOAD en el TRS-80 y se introduce este comando presionando [ENTER] . Esto inicia el movimiento de la cinta. Un * aparece en la línea superior del Monitor; un segundo * parpadea, indicando que el programa se está cargando. Cuando se termine de cargar, el TRS-80 automáticamente apaga el CTR-80 y aparece READY en la pantalla. Entonces se estará listo para ejecutar el programa, para lo cual se teclea RUN y se presiona [ENTER].

2. Para grabar un programa del TRS-80, se presionan las teclas RECORD y PLAY simultáneamente. Después se digita CSAVE en el TRS-80 y se introduce este comando mediante la tecla [ENTER]. Cuando se haya grabado el programa, el TRS-80 apagará automáticamente el CTR-80 y aparecerá READY en la pantalla. Ahora se tiene el programa en cinta (además está aún en el TRS-80). Muchos usuarios de computadora efectúan una segunda y hasta una tercera grabación de la cinta, solamente para asegurarse de tener una buena grabación).

NOTA :

Para cargar los 4 K de memoria (RAM) del TRS-80 no son necesarios más de 3 minutos de cinta de casette.

3. Utilizar el contador de cinta del CTR-80 para ayudarse en la localización de los programas grabados en la cinta.

4. Para obtener mejores resultados, se recomienda el uso de los casettes para computadora de 10 minutos Radio Schack. Pero también se pueden usar cintas standard para audio, asegurándose de usar las de buena calidad, tales como REALISTIC SUPERTAPE. Hay que tener en cuenta, que los casettes para audio tienen zonas, en ambos extremos, de material no magnético y por consiguiente, no se podrá grabar sobre ellas. Se debe avanzar la cinta hasta haber pasado esta zona antes de intentar la grabación.

5. Cuando no se tenga que utilizar el CTR-80 para cargar o grabar programas, no se deben dejar presionadas las teclas RECORD ó PLAY. Pulsar la tecla STOP en estos casos.

6. Para rebobinar (REWIND) o adelantar (FAST-FORWARD) la cinta, se debe desconectar el conector del borne REM; (con el borne REM conectado, el TRS-80 controla el movimiento de la cinta).

7. Si desea guardar un programa grabado en forma permanente, se tiene que romper el protector de borrado del casette (ver el manual del CTR-80).

8. No exponer las cintas grabadas a campos magnéticos, Evitar ponerlas cerca de la toma de c.a.

9. Para comprobar si una cinta tiene un programa grabado, se puede desenchufar el conector del borne EAR (también debe desenchufarse el conector REM para que se pueda controlar el CTR-80 con las teclas) y reproducir la cinta; de esta manera se escucha el programa por el altavoz.

ENCENDIDO DEL SISTEMA

Encender la pantalla de video tirando del botón POWER. Encender el teclado del TRS-80 oprimiendo el botón POWER localizado en la parte posterior-derecha al lado del borne POWER; el diodo luminoso de la derecha del teclado debe encenderse y en la pantalla aparecerá READY
>_ .

Puede ajustarse el contraste y el brillo con los controles localizados en la parte frontal de la pantalla. No se debe colocar el Control de brillo demasiado alto.

NOTA :

Existe un botón de reajuste detrás de una puertecilla situada en la parte posterior izquierda del teclado. Este botón puede usarse para deshacer un estado cíclico de la computadora o bien cuando el TRS-80 no para el casette ó en cualquier situación anormal dentro del programa.

APARTADO 2º

MANUAL DE PROGRAMACION

BASIC NIVEL 1

CAPITULO 1

CARACTERISTICAS DE LAS COMPUTADORAS

Desde el momento en que se enciende, la TRS-80 sigue unas reglas bien definidas para su manejo por el usuario. Esto la convierte en una computadora especialmente fácil de usar. A grandes rasgos, todo lo que se debe hacer es decir lo correcto en el momento adecuado, (a través del teclado). Por supuesto hay "muchas cosas correctas" que se pueden decir; el ensamblarlas para lograr un propósito es lo que se llama programar.

PREPARACION PREVIA A LA PROGRAMACION

1. Conectar el teclado, la pantalla de video y el alimentador de c.c. tal y como se explicó en la sección anterior.

2. Oprimir el botón POWER de la pantalla de video y de la parte posterior del teclado. Esperar unos segundos para que se caliente el tubo de video.

3. Al cabo de unos segundos debe aparecer el mensaje READY

   >_

   en la esquina superior izquierda de la pantalla. Si se oprime la tecla [ENTER] varias veces, la pantalla se llena con mensajes READY, esto quiere decir, que la microcomputadora está preparada para trabajar con ella.

   Para asegurarse de empezar con la computadora despejada, se teclea NEW, para borrar todas las instrucciones de prueba o programas anteriores, y se oprime [ENTER] (no usar la tecla ["SHIFT"], las letras son todas mayúsculas).

La microcomputadora responde borrando la pantalla e imprimiendo READY en la parte superior izquierda de la misma.

>_

4. Para comprobar si la microcomputadora ha encendido correctamente se debe teclear P.M. y oprimir [ENTER]. La pantalla debe mostrar ...

   P.M.

   3583

   Si el número no es 3583, debe apagarse la microcomputadora usando el botón localizado en la parte posterior-derecha del teclado, esperando 1Ø segundos y encendiendo de nuevo. Compruébese que el número es 3583.

   NOTAS :

   - Si se tiene una memoria de 8 K, el número debería ser 7679 y con 16 K se obtendría el número 15871.

   - Es recomendable hacer esta sencilla prueba, cada vez que se encienda el TRS-80. Siempre se teclea NEW [ENTER] antes de efectuar la prueba.

¿ QUE ES UN PROGRAMA DE COMPUTADORA ?

Un programa es una secuencia de instrucciones que la Computadora almacena hasta que se le ordena seguir (o "ejecutar") esas instrucciones. Los programas para el TRS-80 están escritos en un lenguaje llamado BASIC (Beginner's All-purpose Symbolic Instruction Code). Se trata de un lenguaje sencillo y de fácil aprendizaje.

Queremos escribir un programa sencillo de una línea para que la TRS-80 se presente a sí misma. Lo primero que hay que hacer es asegurarse de que la última línea de la pantalla muestra el apuntador > (lo cual indica que la microcomputadora está preparada). Ahora se debe teclear la siguiente línea ...

```
1Ø PRINT" ! HOLA. SOY TU NUEVA MICROCOMPUTADORA TRS-80 ;"
```

¡ No se debe oprimir la tecla [ENTER] todavía !
Si está equivocada la línea, no hay por qué preocuparse; es mucho más fácil la corrección de errores de escritura en la TRS-80, que en una máquina de escribir normal. Simplemente, se debe pulsar la tecla de retroceso (←) y el carácter más a la derecha, se borra. De esta manera se podrá escribir la línea de nuevo.

Observemos atentamente el mensaje escrito por nosotros, respondiendo a las siguientes preguntas :

1. ¿Aparece todo lo que está después de la palabra PRINT dentro de las comillas?
2. ¿Existen algunas otras comillas ?

Si todo está bien, se puede oprimir la tecla [ENTER]. El apuntador (>) reaparecerá. La Computadora está diciendo, "Muy bien, ¿qué sigue?."

Pero ¿ qué sucede si ya es demasiado tarde ?

Si se encuentra algún error después de haber escrito una línea y haber oprimido la tecla [ENTER], no se podrá usar la tecla de retroceso (←) para corregirlo. En su lugar, se reescribe correctamente la línea completa. Tan pronto como se oprima la tecla [ENTER], la nueva línea reemplazará a la incorrecta. Esto se debe a que ambas líneas comparten el mismo número inicial (en este caso, 1Ø).

NOTAS :

- Se observará cómo el pequeño guión (—) se desplaza a través de la pantalla cada vez que se teclea una letra. Este es el "cursor" que informa exactamente en qué lugar se va a imprimir el próximo carácter, oprimiendo la barra espaciadora, se mueve el cursor sin imprimir nada.

- Si se oprime la tecla [ENTER] por segunda vez, en la pantalla se leerá

  READY
  >_

  Esto puede ser alentador, pero no es necesario. Mientras se encuentre el apuntador (>) en la parte inferior de la pantalla, usted sabrá que es "su turno".

EJECUCION DEL PROGRAMA

Ahora se le puede decir a la computadora que ejcute nuestro programa. La orden en BASIC para ello es sencilla : RUN. Así, si se teclea RUN y oprime [ENTER] y no se ha cometido ningún error, la pantalla debe mostrar :

¡ HOLA. SOY TU NUEVA MICROCOMPUTADORA TRS-80 !

Se puede teclear RUN tantas veces como se quiera y en la pantalla aparece siempre la misma respuesta, ya que, cada vez se está procediendo a la ejecución del programa editado.

NOTAS :

- Ya sea que se esté escribiendo un programa, o dando órdenes directas, como RUN, debe oprimirse [ENTER] para informar a la Computadora que se fije en lo que se ha escrito y actúe de acuerdo con ello.

- Es posible "perder el control" de la Computadora, de manera que no aparezca el mensaje de READY cuando se oprima [ENTER]. Para recuperar el control, sólamente hay que oprimir [BREAK], y después [ENTER]. Si ésto no funciona, se debe buscar el botón de Reajuste dentro de la esquina izquierda del TRS-80 y pulsarlo.

Material Aprendido en el Capítulo 1

ORDENES	DECLARACIONES	VARIOS
BREAK	PRINT	> apuntador
ENTER		_ cursor
NEW		← tecla retroceso
RUN		" " comillas

Pondremos una lista como la anterior al final de cada capítulo. Puede usarse como guía para asegurarse de que no se ha olvidado nada.

Probablemente alguien puede preguntarse cual es la diferencia entre las ordenes en BASIC y las declaraciones en BASIC. Las órdenes se ejecutan tan pronto como son tecleadas y se oprime ENTER. Las declaraciones se colocan en los programas y son ejecutadas sólamente después de teclear la orden RUN.

CAPITULO 2

COMO AMPLIAR UN PROGRAMA

Ya se tiene un programa en la Computadora. (Si se apaga entre lecciones, podemos encenderla de nuevo y escribir la línea 1Ø del Capítulo 1). Es tan sólo un programa de una línea, vamos a ampliarlo agregándole una segunda línea. En BASIC, cada línea dentro de un programa debe tener un número, y el programa es ejecutado en orden ascendente, o sea, del número más bajo al número más alto. Podemos escribir :

```
2Ø PRINT "¿LLAMO USTED?  ¿ORDENA ALGO?"
```

Debe verificarse la línea cuidadosamente, especialmente las comillas. Si se ha hecho todo correctamente y se teclea RUN y pulsa [ENTER], la pantalla debe mostrar

```
¡HOLA. SOY TU NUEVA MICROCOMPUTADORA TRS-8Ø ! ¿LLAMO USTED?
¿ORDENA ALGO?
```

Si el programa ha sido ejecutado bien, puede contestar escribiendo :

SI [ENTER]

La pantalla debe mostrar

¿WHAT?

Este mensaje de error es el resultado de un dispositivo interno que busca errores, informa cuando se ha dicho algo incorrecto (o algo correcto en el momento inadecuado). El mensaje WHAT? de la pantalla quiere decir, "No, No, el programa escrito no tiene la información conveniente para que yo pueda dar una respuesta", o palabras similares para tal efecto.

En una lección posterior se verán otros mensajes de error. Entre tanto, si aparece un WHAT? HOW? o un SORRY, debe examinarse el programa en busca del error.

NOTA :

Se ha usado Ø en lugar del número cero, con el fin de que pueda distinguirse la letra del número. De ahora en adelante lo haremos siempre así, a lo largo de este manual.

EL PROGRAMA CRECE

Es costumbre, espaciar las líneas de un programa 1Ø números entre sí. Se puede observar que el programa de dos líneas tiene los números 1Ø y 2Ø. La razón es que es mucho más fácil modificar un programa, si se dejan espacios para intercalar nuevas líneas entre las ya escritas. No se gana nada numerando las líneas más juntas entre sí (como 1,2,3,4) Esto último no debe hacerse nunca.

Escribiendo la nueva línea

15 PRINT [ENTER]

Y haciendo luego

RUN [ENTER]

Debe aparecer :

¡HOLA. SOY TU NUEVA MICROCOMPUTADORA TRS-8Ø !
¿LLAMO USTED? ¿ORDENA ALGO?

¿ Pero que pasa con la línea 15??? Dice PRINT, pero no imprime nada. Eso es lo que sigue a PRINT, y eso es justamente lo que ha impreso. En el proceso de la no impresión, automáticamente se activa el retroceso del carro y se inserta un espacio entre las líneas impresas 1Ø y 2Ø. De esta forma es cómo se inserta un espacio.

Otra declaración importante es REM, que significa REMARK (comentario). Muchas veces es conveniente insertar comentarios en un programa. ¿Por qué? Para poder referirse a ellos más tarde, para ayudar a recordar detalles de programación complicados, o también para saber lo que hace el programa y como usarlo. Es como tener una libreta de anotaciones dentro del programa. Cuando se le dice a la computadora, que ejecute el programa escribiendo RUN y [ENTER], se salta cualquier línea numerada que empiece con la declaración REM. La declaración REM no afecta al programa. Se puede insertar lo que sigue

5 REM *ESTE ES MI PRIMER PROGRAMA DE COMPUTADORA* [ENTER]

luego

RUN [ENTER]

La ejecución del programa no debe verse afectada por la presencia de la nueva línea. Los asteriscos (*) de esta línea son sólo adornos. Cualquier cosa que se escriba después de REM, es ignorada por la microcomputadora.

Observaremos que las líneas añadidas, no ocupan en la pantalla el puesto que les corresponde de acuerdo con su número de línea. Para obtener un listado ordenado de nuestro programa, se debe escribir :

LIST [ENTER]

En la pantalla deberá aparecer :

```
 5 REM * ESTE ES MI PRIMER PROGRAMA DE COMPUTADORA *
10 PRINT  "¡HOLA. SOY TU NUEVA MICROCOMPUTADORA TRS-80!"
15 PRINT
20 PRINT  "¿LLAMO USTED? ¿ORDENA ALGO?"
```

Se puede ordenar un listado (LIST) siempre que aparezca el apuntador en la pantalla.

CUANDO FINALIZA EL PROGRAMA ?

El final del programa es, claro está, la última declaración que se quiere que la computadora ejecute. La mayoría de las computadoras requieren que se ponga una declaración END (Fin), de manera que la computadora sepa que ha terminado. Pero la TRS-8Ø, tiene la declaración END opcional; puede ponerse o no. Hay que recordar, sin embargo, que si se desea ejecutar alguno de estos programas BASIC en otras computadoras, probablemente se necesite la declaración END.

Por las reglas que gobiernan el uso de END, la mayoría de los dialectos del BASIC (que requieren END), insisten en que sea la última declaración en un programa que viene a decirle a la computadora "eso es todo". Por tradición, se le acciona el número 99 ó 999, ó 9999 (ó más grande), dependiendo del número más grande que una determinada computadora acepta. La computadora RADIO SHACK acepta números de líneas hasta 32767.

Podemos agregar una declaración END a nuestro programa, escribiendo :

```
99 END [ENTER]
```

si ejecutamos el programa

```
RUN [ENTER]
```

La pantalla debe mostrar

```
HOLA. SOY TU NUEVA MICROCOMPUTADORA TRS-8Ø !
¿LLAMO USTED? ¿ORDENA ALGO?
```

¿Por qué no se imprime la palabra END? La respuesta es : porque no se imprime nada a menos que sea el "objeto" de una declaración PRINT.

¿Cómo se puede hacer para que la computadora imprima FIN al finalizar la ejecución del programa?

Muy sencillo

```
98 PRINT "FIN"
```

NOTA :

Cuando se llegue a programas más complicados, se debe usar la declaración END para forzar a la computadora a que se pare en los puntos especificados, por eso, END resulta una gran ayuda eún en el TRS-8Ø.

BORRAR SIN REEMPLAZAR

Para mover la declaración END de la línea 99, al número más alto de línea utilizable, 32767, son necesarios dos pasos : primero borrar la línea 99 escribiendo ...

```
99
```

luego [ENTER]

La línea 99 ha sido borrada; pero si se quiere estar seguro, se puede listar el programa completo escribiendo

LIST [ENTER]

En la pantalla deben aparecer las líneas del programa 5, 10, 15, 20 y 98. La 99 no debe estar. Este es el sistema para borrar una línea enteramente.

- El segundo paso es muy sencillo ...

32767 END [ENTER]

... y la nueva línea ha sido introducida. Se puede sacar un listado del programa para ver si ha sido así. Y se puede ejecutar el programa para ver si el traslado de END ha cambiado algo en el mismo. Lógicamente no debe haberlo hecho.

OTROS USOS PARA END

Si se mueve END de la línea 32767 a la línea 17 y se ejecuta el programa con RUN, ¿qué pasa?
Finaliza la ejecución, después de imprimir la línea 1Ø y un espacio.

¿Qué sucede si se mueve END a la línea 13? ¿ Y a la línea 8 ?

Efectuar estas pruebas para entender el efecto de la declaración END, dependiendo del lugar donde está dentro del programa.

OTRO MENSAJE DE ERROR

Vamos a provocar la aparición de un mensaje de error diferente. Para ello traslademos la declaración opcional END de la línea 8 a la línea 5ØØØØ . La computadora contesta con un mensaje de error
HOW?

Está tratando de decir : "Obedezco cualquier orden mientras esté dentro de mis posibilidades". Los números de línea mayores de 32767 están más allá de mi capacidad, así que como, (HOW) voy a obedecer?

NOTA :

En general un mensaje HOW? significa "entiendo las instrucciones pero me piden hacer algo que está fuera de mis posibilidades. "El mensaje de error WHAT?, sin embargo, significa " no entiendo la instrucción, ya sea que esté incorrecta la gramática o que se estén usando palabras que no están en mi vocabulario".

Material Aprendido en el Capítulo 2

ORDENES	DECLARACIONES	VARIOS
LIST	PRINT	Mensajes de error
	REM	WHAT? (¿QUE?)
	END	HOW? (¿COMO?)
		Numeración de las líneas

CAPITULO 3

PUEDE EL TRS-80 EJECUTAR CALCULOS MATEMATICOS ?

La aritmética básica es muy sencilla para el TRS-80. También lo son los cálculos matemáticos complejos, cuando se escriben programas para que los ejecute.
(Estos procedimientos se ampliarán en sucesivos capítulos).

EL BASIC, NIVEL I, utiliza las cuatro operaciones fundamentales de la aritmética, además de una quinta operación que es simplemente una modificación de dos de las anteriores.

1. Adición, usando el símbolo +
2. Substracción, usando el símbolo -
3. Multiplicación, usando el símbolo especial *
4. División, usando el símbolo /
5. Negación (que significa "multiplicación por menos uno"), usando el símbolo _

Hay que tener mucho cuidado para no utilizar una "X" para la multiplicación. La "X" es una letra y nosotros usamos las letras como variables dentro del programa. Por tanto, el asterisco "*" para la computadora es el único símbolo que significa multiplicar.

Por supuesto, también necesitamos el signo "igual" (=). Pero el lenguaje BASIC tiene unas particularidades para la forma en que debe ser usado este signo. Expresiones matemáticas (como 1 + 2*5) pueden expresarse sólamente al lado derecho del signo igual; el lado izquierdo está reservado para la "cantidad variable". Este es el nombre que se da al resultado de la expresión matemática. (todo esto puede parecer un poco extraño, pero es en realidad muy simple, como se descubrirá en las páginas siguientes).

Utilizar todo esto en un programa es sencillo, de modo que vamos a hacerlo. Pero primero debemos borrar el "programa residente" en la memoria de la computadora.
Teclear la orden

NEW [ENTER]

y luego

LIST [ENTER]

para comprobar que no existe nada en la memoria. La computadora debe responder con un simple ...

>_

NOTA :

"Programa residente" es un término usado en las computadoras que quiere decir "lo que hay en la memoria en estos momentos".

HACIENDO TRABAJAR A LA COMPUTADORA

Vamos a utilizar ahora la computadora para resolver problemas sencillos, lo cual significa tener que usar ecuaciones. Una ecuación es tan sólo una pequeña declaración que dice : lo que se encuentra a un lado del signo igual es equivalente a lo que encuentra al otro lado del signo.

Partiendo de la conocida ecuación : "La distancia recorrida es igual a la velocidad multiplicada por el tiempo que dura el viaje", se puede elaborar un programa en BASIC.
Para acortar esta ecuación, podemos escoger algunas letras (llamadas variables) para que representen a estas tres cantidades. Podemos escribir esta ecuación en forma de una declaración aceptable por el TRS-80, y que se encuentra en BASIC.

```
4Ø D = R * T
```

¿Qué hace aquí el 4Ø ? Es el número de la línea del programa. Hay que recordar que todo paso de un programa debe tener su número de línea. Escogemos el 4Ø, pero cualquier otro número puede igualmente valer. Los espacios sobrantes de la línea están ahí sólo para hacer que la ecuación sea más simple de leer; el TRS-80 los ignora. Más adelante, cuando se escriban programas muy largos, probablemente los aprovecharemos debido a que ocupan espacio en la memoria. Por ahora, pueden resultar de mucha utilidad; así es que los dejaremos.

Podemos utilizar cualquiera de las letras del alfabeto, de la A a la Z para identificar los valores que conocemos, así como, aquellos que deseamos calcular. Cuando sea posible, es muy buena idea escoger letras que recuerden aquello que representan, como por ejemplo la D, R y T, que representan la Distancia, la rapidez y el tiempo de la ecuación anterior. Para complicar un poco más este sencillo ejemplo, vamos a ver que existe una manera opcional para escribir esta ecuación, usando la declaración LET de BASIC :

```
4Ø LET D= R * T
```

Este uso de LET, nos recuerda que el haber hecho a D igual a R por T ha sido porque así lo escogemos. En algunas computadoras siempre es necesario el uso de LET al programar ecuaciones. El TRS-80 no lo necesita obligatoriamente, por tanto, se puede o no colocar la declaración LET en la línea.

Esto es lo que la línea 4Ø significa para la computadora. "Toma los valores de R y T, los multiplica y asigna el valor resultante a la variable D. D es igual al resultado de multiplicar R por T.

Podríamos invertir la ecuación y escribir R * T= D
Esto no tendría ningún significado para la computadora. Hay que recordar que, el lado izquierdo de la ecuación está reservado para las variables (no importa la letra que se escoja). El lado derecho es el lugar donde hay que colocar las expresiones matemáticas que involucren números, operaciones y variables conocidas.

Para completar nuestro programa supongamos que :

La distancia (en kilómetros es = a la velocidad, o rápidez, (en kilómetros por hora) multiplicada por el tiempo (en horas). ¿Qué distancia existe de Boston a San Diego, si un avión, viajando a una velocidad media de 5ØØ kilómetros por hora hace el viaje en ó horas ?.

Podemos escribir

```
1Ø REM * DISTANCIA, RAPIDEZ, TIEMPO, PROBLEMA * [ENTER]
2Ø R = 5ØØ [ENTER]
3Ø T = ó [ENTER]
4Ø D = R * T [ENTER]
```

Después de verificar el programa cuidadosamente se teclea ...

```
RUN [ENTER]
```

En la pantalla debe aparecer

```
READY
```

Todo lo que dice es READY ¿ no funciona la computadora ?

Si funciona. La computadora ha multiplicado 5ØØ por ó tal y como le habíamos dicho y ha calculado la respuesta de 3ØØØ kilómetros, pero nos hemos olvidado decirle que nos proporcione la respuesta.

Se recomienda terminar el programa sin ayuda. Tan sólo se necesita una línea más.
Sugerencia : ¿Qué pasa si escribimos 5Ø PRINT "D" ?
Respuesta : No queremos que nos escriba la letra D, sino su valor.

Al final del capítulo se puede encontrar la respuesta a este primer ejercicio.

La respuesta 3ØØØ es correcta, pero su "presentación" no ha sido más espectacular que la respuesta de una calculadora ordinaria de bolsillo. Esto inevitablemente nos lleva de nuevo hacia la declaración PRINT.

Como se ha observado en la línea 5Ø PRINT D, no hay comillas alrededor de la letra D. De todo ésto se puede fácilmente deducir : que si se quieren imprimir PALABRAS EXACTAS en la computadora, éstas deben ir encerradas entre comillas; en cambio, si lo que se quiere es el valor de la variable (en nuestro caso D), hay que omitirlas. Este punto tan sencillo, merece especial atención antes de continuar.

Ahora supongamos que queremos incluir tanto el valor de algo como algunas palabras específicas en la misma línea. Hay que poner mucha atención a la declaración PRINT porque da más problemas a los principiantes que cualquier otro apartado de la programación.
Teclear :

```
5Ø PRINT " LA DISTANCIA (EN KILOMETROS) ES", D  [ENTER]
```

Luego

```
RUN [ENTER]
```

En la pantalla debe aparecer

LA DISTANCIA (EN KILOMETROS) ES 3000

El mensaje incluido entre comillas se ha impreso tal y como lo especificamos y la letra D nos ha dado su valor. La coma le dice a la computadora que queremos que imprima dos cosas separadas, pero en la misma línea. De esta manera se le puede decir que imprima varias cosas separadas, pero en la misma línea, simplemente insertando comas entre ellas.

Tomando como referencia lo que acabamos de decir, ¿cómo se conseguiría que la computadora escribiera el siguiente mensaje?

LA DISTANCIA ES 3000 KILOMETROS.

Muy sencillo, separando los mensajes y la variable D en medio, con comas. Los mensajes deben ir entre comillas.

```
50 PRINT " LA DISTANCIA ES", D, "KILOMETROS".
```

¿Qué sucede con todo el espacio restante de la línea de impresión? La razón de que haya aparecido este espacio, es que la computadora divide el ancho de la pantalla en cuatro zonas de quince caracteres cada una. Cuando una declaración PRINT contiene dos o más elementos separados por comas, la computadora automáticamente imprime los elementos en diferentes zonas de impresión. La distribución automática en zonas es un medio muy conveniente para dar salida a la información en forma tabulada, como veremos más adelante.

Se puede eliminar todo el espacio restante, en el resultado de nuestro programa, sustituyendo las comas (,) por puntos y comas (;), a lo largo de la línea.

Una vez llevado a cabo y tecleando

RUN [ENTER]

La pantalla debe mostrar :

LA DISTANCIA ES 3000 KILOMETROS .

Si se analiza cuidadosamente la línea 50, se aprecia que no existe espacio en blanco entre la S en ES, la D y la K en kilómetros. Pero en la pantalla existe un espacio entre ES y 3000, y otro entre 3000 y kilómetros. La razón es que un punto y coma automáticamente inserta un espacio entre los dos elementos que separa. A medida que se vaya programando, se apreciará la utilidad de este detalle.

Ya hemos analizado instrucciones, declaraciones y operadores matemáticos más que suficientes para solucionar millares de problemas (Operadores matemáticos son los símbolos =,-,+, /,*)
Ahora dediquemos un poco de tiempo a escribir verdaderos programas que solucionen problemas. No hay mejor manera de aprender que haciéndolo, y TODO lo que hemos visto hasta ahora es fundamental para el éxito en los capítulos que siguen. Por tanto, se recomienda no saltarse estos ejercicios; es la mejor manera de entender la programación.
Al final del capítulo se encuentran las respuestas a los ejemplos.

EJERCICIO 3-2 : Escribir un programa que nos diga el tiempo necesario para viajar por avión de San Diego a Boston, si la distancia es de 3000 kilómetros y el avión viaja a 500 KPH.

EJERCICIO 3-3 : Si la circunferencia de un círculo se encuentra multiplicando el diámetro por π (3,14) escribir un programa que encuentre la circunferencia de un círculo cuyo diámetro es de 35 metros.

EJERCICIO 3-4 : Si el área de un círculo se encuentra multiplicando el cuadrado de su radio por π, escribir un programa que encuentre el área de un círculo cuyo radio es de 5 centímetros

EJERCICIO 3-5 : El saldo de su chequera era de $ 225. Ha girado tres cheques (por $17, $35, $225) y realizado dos consignaciones (por $40 y $200). Escribir un programa que ajuste el saldo basado en los cheques girados y las consignaciones hechas, e imprima el nuevo saldo.

Material Aprendido en el Capítulo 3

DECLARACIONES	OPERADORES MATEMATICOS	VARIOS
LET	=	,
	+	;
	-	Variables A - Z
	*	
	/	

SOLUCION A LOS EJERCICIOS PROPUESTOS EN EL 3er. CAPITULO

RESPUESTA TIPO PARA EL EJERCICIO 3-1

```
50 PRINT [ENTER]
```

EJECUCION TIPO PARA EL EJERCICIO 3-1

```
3000
```

NOTA :

Probablemente ha sido utilizado un número de línea diferente en la respuesta pero esto no tiene ninguna importancia; de lo que se trata es de utilizar la declaración PRINT. No hay que desanimarse si no se ha acertado a la primera; en este caso, teclear la línea 50 y ejecutar el programa; luego regresar al 3er. capítulo para continuar.

. . . . . .

RESPUESTA TIPO PARA EL EJERCICIO 3-2

```
10 REM * SOLUCION DEL TIEMPO CONOCIENDO DISTANCIA Y
         RAPIDEZ *
20 D = 3000
30 R = 500
40 T = D/R
50 PRINT "EL TIEMPO REQUERIDO ES "; T; "HORAS".
```

NOTA :

Hay que recordar que se debe oprimir la tecla [ENTER] Después de Cada Línea

EJECUCION TIPO PARA EL EJERCICIO 3-2

```
EL TIEMPO REQUERIDO ES 6 HORAS.
```

NOTA :

Con el fin de llegar a la fórmula de la línea 40 es necesario despejar T de D = R * T

RESPUESTA TIPO AL EJERCICIO 3-3 :

```
10 REM * SOLUCION A LA CIRCUNFERENCIA*
20  P = 3.14
30  D = 35
40  C = P * D
50  PRINT "LA CIRCUNFERENCIA DEL CIRCULO ES"; C;"METROS".
```

EJECUCION TIPO PARA EL 3-3 :

```
LA CIRCUNFERENCIA DEL CIRCULO ES 109.9 METROS.
```

NOTA :

Ya que (π) no está incluida en el BASIC NIVEL I, debemos fijar una variable (en este caso P) igual al valor de pi (3.14)

. . . . . .

RESPUESTA TIPO AL EJERCICIO 3-4 :

```
10  REM * SOLUCION AL AREA CIRCULAR *
20  P = 3.14
30  R = 5
40  A = P * R * R
50  PRINT "EL AREA DEL CIRCULO ES"; A; "CENTIMETROS CUADRADOS".
```

EJECUCION TIPO PARA EL EJERCICIO 3-4 :

```
EL AREA DEL CIRCULO ES 78.5 PULGADAS CUADRADAS.
```

NOTA :

El BASIC NIVEL I, no tiene la función que "eleva a una potencia" para trabajar con R^2. (EL NIVEL 2 sí la trae) en algunos casos sencillos como éste, podemos simplemente usar R por R. (R * R)

RESPUESTA TIPO PARA EL EJERCICIO 3-5 :

No es preciso esforzarse para deducir un programa como el que se indica a continuación : (C = cheques, D= depósitos, B= Balance viejo, N= Balance nuevo).

```
10 B = 225
20 C = 17 + 35 - 225
30 D = 40 - 200
40 N = B-C-D
50 PRINT "SU NUEVO BALANCE ES $"; N
```

CAPITULO 4

OPERATIVIDAD MATEMATICA

A menudo en el mundo matemático es necesario usar cantidades muy grandes o muy pequeñas. En notación normal esto requeriría la escritura de muchos ceros delante o detrás del punto decimal, resultando una cifra de incómodo manejo o inoperante.

Para resolver este problema, la computadora utiliza la "notación exponencial", o "notación científica standard", cuando el tamaño de los números salen de nuestro alcance. El número 5 millones (5,ØØØ,ØØØ) por ejemplo, puede ser escrito "5E + Ø6".
Esto significa, " el número 5, seguido de 6 ceros".

Si obtenemos una respuesta "5E - Ø6", esto significa que debemos mover el punto decimal, que está después del 5, seis lugares hacia la izquierda, poniendo ceros si es necesario. Técnicamente significa 5×10^{-6}, ó 5 millonésimas (ØØØ,ØØ5) . Realmente es muy sencillo cuando se ha entendido el procedimiento, ya que facilita la localización de los números, sin perder la posición del punto decimal. Debido a que la computadora insiste en que se use este método para números muy grandes o muy pequeños, es bueno que nos vayamos formando el hábito.

Se recomienda ejecutar los siguientes ejercicios para acostumbrarse al uso de esta notación, pero antes teclear NEW.
De ahora en adelante no indicaremos el uso de la tecla [ENTER] por considerarlo suficientemente conocido para el lector.

EJERCICIO 4-1 : Si un millón de automóviles recorrieron diez mil millas cada uno en cierto año, ¿cuántas millas recorrieron entre todos ese año? Escribir y ejecutar un programa sencillo que conteste la pregunta.

EJERCICIO 4-2 :Cambiar las líneas 2Ø y 3Ø del programa para la Solución de las Millas recorridas por Automóviles (del ejercicio 4-1) para que exprese los números en notaciones exponencial o SSN (Standard Scientific Notation).

NOTA :

(La E representa "exponente", y en nuestro caso se refiere al exponente de 1Ø, esto es, los ceros a la derecha, o a la izquierda, del número principal).

Material Aprendido en el Capítulo 4

VARIOS

Notación - E

SOLUCION A LOS EJERCICIOS PROPUESTOS EN EL 4º CAPITULO

RESPUESTA TIPO PARA EL EJERCICIO 4-1 :

```
1Ø REM *PROGRAMA PARA LA SOLUCION DE LAS MILLAS DE LOS
   AUTOMOVILES *
2Ø  N = 1ØØØØØØ
3Ø  D = 1ØØØØ
4Ø  T = N * D
5Ø  PRINT " EL  NUMERO TOTAL DE MILLAS RECORRIDAS ES", T
```

EJECUCION TIPO PARA EL EJERCICIO 4-1 :

```
EL NUMERO TOTAL DE MILLAS RECORRIDAS ES 1E + 1Ø
```

NOTA :

Como se explicó anteriormente, esta respuesta es el número 1 seguido de 1Ø ceros. La computadora no imprime ningún número mayor que 999,999 sin convertirlo a notación exponencial.

```
1Ø,ØØØ,ØØØ,ØØØ
```

. . . . . .

RESPUESTAS TIPO PARA EL EJERCICIO 4-2 :

```
2Ø N = 1E + 6
3Ø D = 1E + 4
```

EJECUCION TIPO PARA 4-2 :

```
EL NUMERO TOTAL DE MILLAS RECORRIDAS ES 1E + 1Ø
```

NOTA :

La respuesta coincide con la obtenida anteriormente, lo cual significa que no tan sólo recibimos las respuestas en SSN, sino que también, podemos utilizarlas en nuestros programas.

CAPITULO 5

USO DEL PARENTESIS Y ORDEN EN LAS OPERACIONES

Los paréntesis juegan un importante papel dentro de la programación de computadoras, lo mismo que en las matemáticas ordinarias. Se utilizan aquí de la misma manera, pero existen excepciones importantes.

1. En BASIC, los paréntesis pueden encerrar las operaciones que deben ser efectuadas. Aquellas operaciones que están dentro del paréntesis se efectuarán antes que las que están fuera del paréntesis.

2. Aquellas operaciones que se encuentren en el interior de los paréntesis (esto es, paréntesis dentro de paréntesis) serán realizadas primero.

3. Cuando exista un "empate", en cuanto a qué operaciones debe efectuar primero la computadora después de haber eliminado los paréntesis, va avanzando a lo largo de la línea del programa de izquierda a derecha efectuando las multiplicaciones y divisiones.
   Después comienza de nuevo por la izquierda y efectúa las adiciones y substracciones.

   NOTA :

   Las funciones INT, RND y ABS, se efectúan antes que la multiplicación y división. (No se ha hablado aún de esto, pero lo decimos para completar el apartado).

4. Un problema que presenta la multiplicación : XY no se interpreta bien por la computadora, si no se coloca el signo de multiplicar * . Así pues, debe escribirse X * Y.

NOTAS :

- Si se desea estar seguro de que se calculen correctamente los problemas, hay que usar ( ) alrededor de las operaciones que se desee efectuar primero.
- En matemáticas se supone que se debe efectuar la Multiplicación y la División primero (de izquierda a derecha), luego regresar para la Adición y la Substracción (de izquierda a derecha). El TRS-80 utiliza la misma secuencia.

Vamos a practicar el uso de los paréntesis mediante un ejemplo :

Para convertir la temperatura Fahrenheit a Celsius (centígrados), se utiliza la siguiente relación :

La temperatura Fahrenheit es igual a 32 grados centígrados más nueve quintos de la temperatura en grados centígrados.

$$F^{o} = \frac{9}{5} \times C^{o} + 32$$

Supongamos que tenemos una temperatura Celsius de 25º. Escribir y ejecutar el siguiente programa :

```
1Ø REM *CONVERSION DE CELSIUS A FAHRENHEIT *
2Ø  C = 25
3Ø  F = (9/5) * C + 32
4Ø  PRINT C; "GRADOS CELSIUS ="; F; "GRADOS FAHRENHEIT"
```

La pantalla debe mostrar

```
25 GRADOS CELSIUS = 77 GRADOS FAHRENHEIT
```

Observar primero, que la línea 4Ø está formada por una declaración PRINT seguida de cuatro expresiones distintas, dos variables y dos conjuntos de palabras entre comillas llamadas "literales" o "condiciones".

Segundo : obervar cómo se han colocado los paréntesis en la línea 3Ø. Con el 9/5 asegurado dentro del paréntesis, podemos multiplicar su cociente por C, y luego agregar 32.

Ahora se puede eliminar el paréntesis de la línea 3Ø y ejecutarlo de nuevo. La respuesta permanece invariable ¿Por qué?

1. En el primer paso, la computadora comenzó por realizar todas las operaciones que estaban entre paréntesis, en este caso, sólo una (9/5). Calculó, pero no imprimió, 1.8. Luego multiplicó 1.8 por el valor de C y le agregó 32.

2. En nuestro siguiente intento, sin los paréntesis, la computadora simplemente se movió de izquierda a derecha efectuando primeramente el problema de la división (9 dividido entre 5), luego la multiplicación (1.8 por C), después la adición (agregar 32). En este primer ejemplo, los paréntesis podían haber sido omitidos.

¿Qué sucede si se cambia + 32 por 32+ y se coloca al comienzo de la ecuación de la línea 3Ø ? Ejecutar de nuevo el programa sin los paréntesis.

¿Por qué no cambió la respuesta?
Solución : La ejecución se realiza de izquierda a derecha, primero la multiplicación y la división, luego vuelve y realiza la adición y substracción.
Por este motivo es por lo que el 32 no se agrega al 9 antes de dividirse entre 5. ¡MUY IMPORTANTE!. Si hubieran sido sumados, habríamos obtenido una respuesta incorrecta.

EJERCICIO 5-1 : Escribir y ejecutar un programa que convierta 65° Fahrenheit a Celsius. La regla nos dice que "la temperatura en Celsius es igual a nueve quintos de lo que sobre a la temperatura Fahrenheit, después de haberle restado 32° ".

$$C^\circ = (F^\circ - 32) \times \left[\frac{5}{9}\right]$$

EJERCICIO 5-2 : Quitar el primer juego de paréntesis en la respuesta del Ejercicio 5-1 y ejecutarlo nuevamente (RUN).

EJERCICIO 5-3 : Reemplazar el primer juego de paréntesis en la línea número 3Ø del programa y quitar el segundo par de paréntesis, luego ejecutarlo (RUN). Observar cómo la respuesta se obtiene correctamente.

EJERCICIO 5-4 : Poner paréntesis en la siguiente ecuación para que sea correcta. Escribir un programa que se pueda comprobar en el TRS-80

30 - 9 - 8 - 7 - 6 = 28

Material Aprendido en el Capítulo 5

VARIOS
( )
Orden de las Operaciones

SOLUCION A LOS EJERCICIOS PROPUESTOS EN EL CAPITULO 5º

RESPUESTA TIPO AL EJERCICIO 5-1 :

```
1Ø REM *CONVERSION FAHRENHEIT A CELSIUS*
2Ø F = 65
3Ø C = (F - 32) * (5/9)
4Ø PRINT F; "GRADOS FAHRENHEIT ="; C; "GRADOS CELSIUS".
```

EJECUCION TIPO PARA EL EJERCICIO 5-1:

```
65 GRADOS FAHRENHEIT = 18.3333 GRADOS CELSIUS
```

Observar cuidadosamente, cómo se han puesto los paréntesis. Por regla general, cuando se tenga duda, se deben usar paréntesis; lo peor que puede suceder en este caso es que la computadora aminore la velocidad del cálculo en unos cuantos microsegundos.

. . . . . .

RESPUESTA TIPO AL EJERCICIO 5-2 :

```
3Ø C = F - 32 *(5/9)
```

EJECUCION TIPO PARA EL EJERCICIO 5-2 :

```
65 GRADOS FAHRENHEIT = 47.222 GRADOS CELSIUS
```

. . . . . .

RESPUESTA TIPO AL EJERCICIO 5-3 :

```
3Ø C = (F - 32) * 5/9
```

EJECUCION TIPO PARA EL EJERCICIO 5-3 :

```
65 GRADOS FAHRENHEIT = 18.3333 GRADOS CELSIUS
```

RESPUESTA TIPO PARA EL EJERCICIO 5-4 :

Hay dos respuestas posibles : 3Ø - (9-8) - (7-6) = 28

3Ø - (9 -(8-(7-6)))= 28

Programa tipo :

```
1Ø  A = 3Ø - (9 - (8 - (7 - 6)))
2Ø  PRINT A
```

O la línea 10 podría ser

```
A = 3Ø - (9 -8) - (7-6)
```

CAPITULO 6

DECLARACIONES DE TOMA DE DECISION

Las declaraciones IF - THEN y GOTO (SI- ENTONCES) y (VETE A) permiten que la computadora tome decisiones. Pero primero veamos unos cuantos operadores más ...

Los Operadores de Relación permiten a la Computadora comparar un valor con otro. Existen sólamente tres :

1. Igual, usando el símbolo =
2. Es mayor que, usando el símbolo >
3. Es menor que, usando el símbolo <

Combinando estos tres símbolos, obtenemos tres operadores más :

4. No es igual a, usando el símbolo <>
5. Es menor que o igual a, usando el símbolo < =
6. Es mayor que o igual a, usando el símbolo > =

Sumando estos seis operadores de relación a los cuatro operadores matemáticos que ya conocemos, más las nuevas DECLARACIONES, llamadas IF-THEN, & GO TO, se crea un sistema muy poderoso de comparación y cálculo que constituye la médula de todo lo que sigue.

La declaración IF-THEN combinada con los seis operadores de relación anteriores, nos da la parte activa de un sistema lógico.
Escribir y ejecutar el siguiente programa :

```
1Ø  A = 5
2Ø  IF A = 5 THEN 5Ø
3Ø  PRINT " A NO ES IGUAL A 5 "
4Ø  END
5Ø  PRINT " A IGUAL A 5"
```

En la pantalla debe aparecer :

A IGUAL A 5.

Vamos a examinar el programa línea por línea.

La línea 1Ø establece que A tiene un valor de 5.
La línea 2Ø es una declaración IF-THEN la cual le dice a la computadora que vaya a la línea 5Ø SI EL VALOR ES EXACTAMENTE 5, saltando sobre todo aquello que se encuentre entre las líneas 2Ø y 5Ø. Como A es igual a 5, la computadora salta a la línea 5Ø y hace lo que allí se le indica : imprimir A IGUAL A 5. Las líneas 3Ø y 4Ø no se usan para nada en este caso.

Si se cambia la línea 1Ø por :

```
1Ø  A = 6
```

y se ejecuta de nuevo el programa, la pantalla debe mostrar :

A NO ES IGUAL A 5

Vamos a analizarlo línea por línea.

La línea 1Ø establece que el valor de A es 6.

La línea 2Ø prueba el valor de A. Si A es igual a 5 THEN (ENTONCES) la computadora va a la línea 5Ø . Pero "si falla la prueba", ésto es, si A NO es igual a 5, la computadora prosigue normalmente hacia la línea siguiente, o sea, la línea 3Ø.

La línea 3Ø indica a la computadora que debe imprimir el hecho de que A NO ES IGUAL A 5. No nos indica cual es el valor de A, tan sólo que no es igual a 5. La computadora prosigue hacia la línea siguiente.

La línea 4Ø finaliza la ejecución del programa (END). Sin esta declaración que separa las líneas 3Ø y 5Ø, la computadora seguiría hacia la línea 5Ø e imprimiría su contenido, el cual, obviamente está en contradicción con el contenido de la línea 3Ø. Este es un ejemplo del uso de la declaración IF-THEN con el operador de relación más fundamental, el signo igual.

Veamos si se puede lograr lo mismo utilizando el signo "NO ES IGUAL" :

EJERCICIO 6-1 : Escribir de nuevo el programa residente usando el signo "no es igual a" en la línea 2Ø en lugar del signo igual, modificando las demás líneas según sea necesario, de manera que se obtengan los mismos resultados con el nuevo programa y el del ejemplo.

EJERCICIO 6-2 : Cambiar la línea 1Ø para darle a A el valor 6. Dejando las otras cuatro líneas del ejercicio como se muestran, agregar más líneas al programa (si son necesarias) con el fin de averiguar si A es mayor o menor que 5. Ejecutar el programa cambiando el valor de A en la línea 1Ø para estar seguros de que funciona.

La declaración IF-THEN es lo que se conoce como bifurcación CONDICIONAL. El programa se dirigirá a otro lugar del mismo con la condición de que pase la prueba que contiene la declaración. Si falla la prueba, el programa simplemente continúa hacia la siguiente línea. Una declaración llamada GOTO es conocida con frecuencia como bifurcación INCONDICIONAL. Si se reemplazaran las líneas 4Ø y 8Ø con GOTO 99 y se añadiera la línea 99 :

```
99 END
```

sucedería, que en el momento en que se pasara por la línea 4Ø o por la 8Ø incondicionalmente, el programa saltará a la línea 99, que en este caso es el final de la ejecución. Se recomienda practicar estos cambios y ejecutar el programa, para comprender mejor la declaración GOTO.

NOTA :

No se debe dejar espacios entre GO y TO (algunas variaciones del BASIC lo permiten, el TRS-80 requiere utilizar una sola palabra.

Material Aprendido en el Capítulo 6

DECLARACIONES	OPERADORES DE RELACION	VARIOS
IF-THEN	=	Bifurcación Condicional
GOTO	>	Bifurcación Incondicional
	<	
	<>	
	<=	
	>=	

SOLUCION A LOS EJERCICIOS PROPUESTOS EN EL CAPITULO 6º

RESPUESTA TIPO PARA EL EJERCICIO 6-1 :

```
1Ø  A = 5
2Ø  IF A<>5 THEN 5Ø
3Ø  PRINT " A IGUAL A 5"
4Ø  END
5Ø  PRINT "A NO ES IGUAL A 5"
```

EJECUCION TIPO PARA EL EJERCICIO 6-1 :

```
A IGUAL A 5

      . . . . . .
```

RESPUESTA TIPO PARA EL EJERCICIO 6-2 :

```
1Ø  A = 6
2Ø  IF A<>5 THEN 5Ø
3Ø  PRINT " A IGUAL A 5"
4Ø  END
5Ø  PRINT "A NO ES IGUAL A 5"
6Ø  IF A<5 THEN 9Ø
7Ø  PRINT " A ES MAYOR QUE 5"
8Ø  END
9Ø  PRINT "A ES MENOR QUE 5"
```

EJECUCION TIPO PARA EL EJERCICIO 6-2 :

```
A NO ES IGUAL A 5
A ES MAYOR QUE 5
```

NOTA :

Hemos tenido que poner otra declaración END (línea 8Ø) para evitar que el programa pasara a la línea 9Ø después de imprimir la línea 7Ø

CAPITULO 7

USO DE LA DECLARACION INPUT PARA COMUNICARSE CON EL TRS-80

Vamos a comenzar este capítulo tecleando el programa ejemplo del ejercicio 6-2 :

Resulta molesto tener que escribir de nuevo la línea 1Ø, cada vez que se desea cambiar el valor de A. Existe, sin embargo, una declaración en BASIC, que permite introducir datos a la hora de ejecutar el programa. Esta es la declaración INPUT, que constituye una manera más rápida y sencilla de lograr lo mismo.

Podemos agregar las siguientes líneas al programa residente :

```
 5 PRINT " EL VALOR QUE DESEO DAR ES"
1Ø INPUT A
```

Al ejecutar el programa

La computadora debe imprimir :

```
EL VALOR QUE DESEO DAR ES
?_
```

El signo de interrogación de la pantalla, significa, "estoy esperando a que se introduzca el dato". Si se teclea un número, debe suceder lo mismo que en el capítulo anterior. Ejecutar varias veces el programa para practicar con la declaración INPUT.

Se le puede dar un toque de elegancia al procedimiento volviendo a escribir la línea 5 del siguiente modo :

```
5 PRINT " EL VALOR QUE DESEO DARLE A A ES";
```

Como puede apreciarse, se le ha añadido un punto y coma al final de la línea.

En capítulos anteriores, usábamos los puntos y comas en las declaraciones PRINT, pero sólamente como medio para unir varias de ellas, de manera que se imprimieran más cerca una de otra en la misma línea. En este caso, hemos puesto un punto y coma al final, de manera que el signo de interrogación de la línea siguiente se imprima en la misma línea, en lugar de hacerlo debajo. Después de cambiar la línea 5 como hemos visto anteriormente y ejecutado el programa, nos debe aparecer en la pantalla :

EL VALOR QUE DESEO DARLE A A ES ?

Observar, que no se puede usar el punto y coma indistintamente al final de una declaración PRINT. Sólo se hace para unir dos líneas, que deben imprimirse. La línea INPUT imprime el signo de interrogación. Más tarde veremos cómo pueden unirse dos líneas que empiezan por PRINT haciendo uso del punto y coma.

El BASIC del TRS-80, permite una forma todavía más sencilla de introducir el valor de A. Es, por tanto, la más usada.

Si escribimos de nuevo la línea 5 para que diga :

```
5 INPUT " EL VALOR QUE DESEO DARLE A A ES"; A
```

y suprimimos la línea 1Ø, al ejecutar el programa (RUN), obtendremos los mismos resultados .

Observemos los cambios realizados :

1. Sustitución de PRINT por INPUT
2. Escritura de las dos declaraciones en la misma línea.
3. Eliminación de la línea extra.

En programas largos, esta forma de proceder es de gran utilidad.

Hasta ahora, todos los programas, han sido estrictamente directos. Se teclea RUN, la computadora ejecuta el programa, imprime los resultados (si existen) y aparece READY. Para repetir el programa, se tiene que teclear RUN nuevamente.

¿Existe algún otro método para hacer que la computadora ejecute un programa dos o más veces? Desde luego siempre cabe la posibilidad de escribirlo dos veces, pero hay una forma mucho más sencilla. Se recomienda intentar averiguarla por uno mismo antes de seguir leyendo.

La solución a esta pregunta se encuentra en el programa siguiente : (Antes de teclearse se debe oprimir la tecla NEW para borrar la memoria).

```
10  REM *PROGRAMA MEJORADO PARA LA CONVERSION CELSIUS
    A FAHRENHEIT *
20  INPUT " CUAL ES LA TEMPERATURA EN GRADOS CELSIUS"; C
30  F = (9/5)* C + 32
40  PRINT C; "GRADOS CELSIUS ="; F; "GRADOS FAHRENHEIT".
50  GOTO 20
y RUN
```

La computadora sigue preguntando datos y respondiendo hasta que se la apague. Esta es la gran característica de las computadoras : hacer algo una y otra vez. Se pueden modificar algunos de los programas anteriores para hacerlos auto-repetitivos.

NOTA :

No es necesario apagar la computadora para salir del ciclo, basta con apretar la tecla BREAK

Material Aprendido en el Capítulo 7

Declaraciones	Varios
INPUT	; Final
y	punto y coma de fin
INPUT utilizado conjuntamente con PRINT	de línea

Hemos visto hasta aquí 7 interesantes lecciones. Se recomienda su repaso antes de continuar.

CAPITULO 8

MODALIDAD DE CALCULADORA

Antes de continuar explorando nuestra computadora -trabajando como computadora debemos observar que también trabaja como calculadora. Si se omite el número de la línea antes de ciertas instrucciones, la computadora las ejecuta, dando la respuesta en la pantalla. Además, trabaja como calculadora, aún cuando esté cargada con un programa, realizando las operaciones ó prescindiendo de dicho programa. Todo lo que se necesita, para estar en la modalidad de calculadora es tener en pantalla el apuntador >.

Ejemplo : ¿Cuánto es 3 por 4? Escribimos :

```
PRINT  3 * 4
```

... La respuesta aparece en la pantalla :

12

Ejemplo : ¿Cuánto es 345 dividido entre 123?

```
PRINT  345/123
```

... la respuesta es

2.8Ø488

Invertir unos minutos elaborando problemas aritméticos rutinarios, usando la modalidad de calculadora. Cualquier expresión aritmética, que pueda usarse en un programa, puede también evaluarse en la modalidad de calculadora. Esto incluye paréntesis y cálculos en cadena como A* B *C.

ERROR DE CALCULO EN LA TRS-80

Efectuar la siguiente operación :

PRINT (2/3)* (3/2)

La respuesta sobre la pantalla es :

1.ØØØØØØ9

Un número multiplicado por su recíproco se supone que debe ser igual a uno (1) exactamente.
¿Qué es lo que sucede? Hemos descubierto el límite de precisión de la computadora. Igual que en las calculadoras, una computadora no puede ser totalmente precisa.

Para expresiones aritméticas cortas, la TRS-80 es precisa hasta el quinto o sexto lugar decimal. En expresiones más largas y más complejas, un pequeñísimo error en el sexto lugar decimal puede ser amplificado hasta hacerse importante. Todos los programadores deben acostumbrarse a este tipo de error incorporado a la computadora. Discutiremos la forma de poderlos manejar en un capítulo posterior.

MODALIDAD DE CALCULADORA PARA DETECTAR ERRORES

Supongamos que un programa, no está dando la respuesta que esperábamos.
¿Cómo podemos verificar el programa? Una forma es preguntarle a la computadora qué sucede, hasta donde ella conozca, acerca de las variables usadas en su programa residente.

Si en un programa que utiliza como variable la letra X, tecleamos :

PRINT X

la computadora dirá cual es el valor actual de X.

NOTA :

Siempre hay algo almacenado en cada celda de memoria de la computadora, aún cuando no se haya puesto nada ahí. Introducir y ejecutar esta instrucción

```
PRINT  A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z
```

Las respuestas dependen de los últimos valores dados a esas variables, aunque sean de programas anteriores. Si se ha apoyado la computadora desde la última vez que se usaron algunas de estas variables, los números almacenados en estas posiciones deben ser totalmente arbitrarias.

LA INSTRUCCION DE MEMORIA

Ya que los programas que se escriben ocupan espacio en la memoria de la computadora, y el tamaño de un programa está limitado a la cantidad de memoria que se tenga, puede ser importante saber cuanta memoria se está usando para un programa dado. Para averiaguarlo se usa la instrucción de memoria.

La TRS-80 viene provista de memorias de distintas capacidades, el menor de las cuales es 4 K. Esto quiere decir, que hay cerca de 4ØØØ diferentes posiciones para almacenar y procesar los programas (en realidad 4Ø96). (Si se tiene una memoria de 8K, las posiciones son 8192; y para 16 K son 16384).

Para enternder mejor lo que esto significa, se puede escribir :

```
NEW
PRINT MEM
```

En la pantalla debe aparecer :

```
3583
```

NOTA :

Si no se obtiene esta respuesta (por ejemplo, un número muy grande o uno que contenga un signo menos (-), se debe apagar la computadora durante diez segundos y volver a encender; se teclea de nuevo PRINT MEM. Si el TRS-80 tiene más RAM, se debe obtener un número mayor, tal como los siguientes :

8 K RAM = 7679

16 K RAM = 15871

Sin ningún programa cargado, existen 3583 posiciones de memoria disponibles para ser usadas. La diferencia en espacio de memoria entre 3583 y 4096 se reserva para procesar programas, gestionar y "controlar" lo que está haciendo la computadora.

Vamos a escribir este sencillo programa :

```
1Ø  A =  25
```

Ahora medimos la memoria restante ...

```
PRINT MEM
```

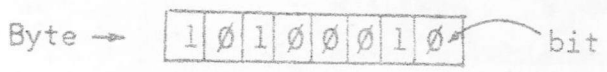
La respuesta debe ser ...

```
3573
```

El programa ha tomado, 3583 - 3573 = 1Ø bytes de espacio de memoria.

NOTA :

Vamos a dar una rápida explicación de lo que es un byte. El término byte se emplea en teoría de microprocesadores para designar un grupo de 8 bits, siendo cada bit una célula de memoria que almacena un 1 ó un Ø lógicos.



La combinación de unos y ceros lógicos componen, de acuerdo con un determinado código, la información que se teclea, así como, el tratamiento interno de dicha información por el ordenador. Se recomienda no perder mucho tiempo en analizar con detalle esta nota.

Ahora vamos a descomponer el empleo de los espacios de memoria de nuestro ejercicio.

1. Cada número de línea (sin importar que el número sea grande o pequeño), y el espacio que le sigue ocupa 3 bytes de memoria. El retorno del carro al final de la línea ocupa otro byte, aunque no se imprima nada. Así, la utilización de memoria para cualquier línea es siempre de cuatro bytes, más los bytes que utilice el texto impreso.

2. Cada letra, número o espacio ocupa 1 byte. En nuestro programa ocupamos entonces 6 bytes.

En total el número total de bytes empleados es 4 + 6 =10 bytes.

Introducir la siguiente línea adicional, dejar la línea 1Ø tal y como está y calcular la cantidad de espacio que sobra en la memoria comparándola con la obtenida mediante la instrucción PRINT MEM.

```
2Ø PRINT "ESTE EJEMPLO ES PARA MEDIR EL USO DE LA MEMORIA".
```

¿Cuánto espacio queda en la memoria?

Respuesta : La línea 1Ø toma 1Ø bytes. La línea 2Ø toma cuatro bytes por el "servicio" más 48 caracteres = 52 bytes. 52 + 1Ø = 62 bytes. 3583 - 62 = 3521 bytes.

Teclear :

PRINT MEM

para ver si es correcto el resultado.

Obviamente los programas cortos de aprendizaje que hemos estado escribiendo hasta ahora, no necesitan mucho espacio de memoria. Esto cambia rápidamente, sin embargo, a medida que nos adentramos en una programación más sofisticada, es interesante escribir PRINT MEM cuando se completa un programa, porque desarrolla el sentido de su tamaño.

OTRO MENSAJE DE ERROR

Hasta ahora hemos visto dos tipos de mensajes de error :

WHAT? - No entiendo que es lo que se quiere.

HOW? - Entiendo lo que se me está diciendo pero no puedo hacerlo.

A continuación vamos a ver el tercer y último mensaje de error, que viene indicado por la palabra :

SORRY

Significa "Lo siento se han acabado las posiciones de memoria y se debe recortar el tamaño del programa o comprar memoria adicional".

Con algo de práctica se puede predecir cuánta memoria ocupa un programa dado. Todas las lecciones y programas que se incluyen en el sistema TRS-80 NIVEL I entran en los 4 K de memoria disponibles.

Material Aprendido en el Capítulo 8

Instrucciones	Varios
PRINT MEM	Modalidad de Calculadora
	Memoria
	Byte
	SORRY

CAPITULO 9

USO DEL CASETTE DE CINTA

Pronto se escribirán programas largos y potentes. Resulta en extremo difícil y pesado editarlos sin errores y resultaría inoperante tener que realizar todo este trabajo, cada vez que se necesitara ejecutarlos. Esto último explica la necesidad del uso del casette de cinta.

El TRS-80 posee una tarjeta adaptadora (interface) para grabadora de cinta pudiéndose grabar y almacenar en ésta cualquier programa en cintas casettes de alta calidad. Los "4K" de la memoria pueden ser "almacenados" en la cinta procedentes de la memoria de la computadora o "cargados" de la cinta a la memoria en tres minutos. La mayoría de programas son más cortos y requieren aún menos tiempo. La gran característica del uso del casette de cinta es que permite formar una biblioteca de programas propios para la computadora. De este modo se pueden intercambiar programas con otros usuarios de la TRS-80.

Al comienzo de este manual, se habló del procedimiento de conexión y uso de la grabadora. En este capítulo, se van a explicar detenidamente los pasos para el almacenamiento (en la casette) y carga (en la memoria de la computadora) de programas.

GRABACION

Sólo es necesario un poco de práctica :

1. Localizar la grabadora, el cable de conexión y el casette de cinta para grabaciones de computadora Radio Shack.

2. Conectar el cable corto entre el borne TAPE localizado en la parte posterior del teclado de la TRS-80 y la grabadora de casette :

   A. El conector gris de clavija delgada, va al borne REM de la grabadora.

   B. El conector gris de clavija más gruesa, va al borne AUX.

   C. El conector negro va al borne EAR.

3. Conectar la grabadora a la red (ó ponerle baterías).

4. Editar cualquier programa en la computadora, preferentemente uno que tenga varias líneas. Ejecutarlo (RUN) para asegurarse de que está correctamente escrito.

5. Cargar el casette de cinta y oprimir los botones PLAY y RECORD al mismo tiempo hasta que queden fijos.

6. Almacenar el programa en la cinta escribiendo la instrucción :

   CSAVE (que significa "guardar en la casette")

De esta manera el motor de la grabadora comienza a funcionar y se graba el programa en la cinta.

Cuando aparece en la pantalla

```
READY
>_
```

entonces se para el motor, el programa ha quedado grabado en la cinta. Además se encuentra aún en la memoria de la computadora. Tan sólo ha sido "copiado".

NOTA :

Se recomienda colocar el "conector falso" dentro del borne MIC (esto evita que se introduzcan ruidos por el micrófono interno cuando se graban programas).

7. Si se desea, se puede efectuar una pequeña prueba :
Desconectar el pequeño conector del borne REM de la grabadora y rebobinar la cinta. Si se reproduce la cinta, desenchufando el conector negro del borne EAR, se puede oír el sonido de la información digital.

CARGA

El proceso inverso al anterior, o sea, la carga (copia) del programa de la cinta a la computadora, es sencillo :

1. Asegurarse de que la cinta se rebobina completamente, (hasta el comienzo del programa), y que los conectores están en su lugar.
2. Oprimir el botón PLAY hasta que quede fijo. Poner el control de volumen entre 7-8.
3. Teclear NEW para borrar cualquier programa existente.
4. Teclear la instrucción

   CLOAD (que significa "cargar desde el casette")

   El motor de la grabadora comienza a funcionar y la información fluye de la cinta a la memoria de la computadora a razón de aproximadamente 12ØØ bytes por minuto.

   Tan pronto como la computadora detecta la información, aparece un asterisco (*) en la pantalla; luego a medida que acepta cada línea de información, un segundo (*) asterisco parpadea.

   Cuando aparezca

   READY
   >_

   el programa habrá sido cargado y el motor de la grabadora se pararó.

5. Se puede ejecutar el programa, para comprobar que la transferencia de datos ha sido correcta. En el caso poco frecuente de que ésto no sucediera, hay que repetir los pasos anteriores, asegurándose de que todos los cables están conectados correctamente, el volumen colocado entre 7-8 y que las cabezas de la grabadora estén limpias. Puede oírse la cinta para comprobar.

NOTA :

Si la grabadora no para, abrir la puertecilla situada en la parte posterior izquierda del teclado y oprimir el botón de reajuste. Esto devuelve el control al teclado, el cual está en esos momentos en la computadora bajo la modalidad CLOAD ó CSAVE.

INFORMACION SOBRE LA CINTA

Para minimizar la posibilidad de encontrar un "punto blando" en la cinta, en el oxido, los operadores con experiencia realizan un "doble almacenado" al copiar de la computadora a la cinta. Esto quiere decir, que se debe copiar dos veces el programa sobre la misma cinta - una grabación después de otra. En vaciados largos, uno se realiza en una dirección y el otro en la otra dirección. Para mayor seguridad, especialmente en programas importantes se acostumbra grabarlos en más de una cinta. Los errores son poco frecuentes, pero la experiencia es, después de todo, la mejor consejera.

La cinta de computadora Radio Shack es de calidad uniforme , escogida especialmente por sus características de baja distorsión. Si tan sólo se pierde un pequeño bit se pierde el programa entero.

Esta cinta viene enrollada en carretes más cortos que lo normal, siendo standard el C-10 que graba cinco minutos en cada dirección; mucho más que suficiente para la mayoría de los programas. La memoria completa de 4K puede ser grabada sobre un lado del C-10, quedando el otro lado disponible para otro programa sin necesidad de tener que hacer largos rebobinados.

Los profesionales con experiencia en computadoras han encontrado que es mejor usar un casette separado (o por lo menos en lado separado) para cada programa, en lugar de tratar de buscar a lo largo de cintas largas el programa deseado.

Cuando no se usa la Grabadora para cargar o grabar, no hay que dejar oprimidas las teclas RECORD y PLAY (oprimir STOP).

No se deben exponer las cintas grabadas a campos magnéticos. Hay que evitar ponerlas cerca de las tomas de corriente.

No se debe intentar grabar de nuevo sobre una cinta grabada anteriormente, ya que aún cuando el nuevo proceso de grabación borra el programa viejo, puede quedar suficiente información para confundir la nueva grabación. Si se quiere usar la misma cinta una segunda o tercera vez, hay que utilizar un borrador de cintas de alta calidad para asegurarse de que la información vieja ha sido totalmente borrada.

Si se quiere guardar permanentemente un programa en cinta, debe romperse el protector de borrado del casette (véase el Manual de la Grabadora). Cuando los protectores hayan sido rotos, no se podrá oprimir la tecla RECORD de la grabadora (esto evitará borrar accidentalmente la cinta).

RETROALIMENTACION A TIERRA

En algunas grabadoras, si se dejan conectados los bornes del audífono y el AUX al mismo tiempo, cuando se hace una grabación, se termina con un pequeño zumbido dentro del programa. Esto es debido a una retroalimentación a tierra entre la grabadora y los cables. Para evitar problemas de retroalimentación a tierra, hay que mantener conectado sólamente el audífono ó el auxiliar, PERO NO AMBOS.

Material Aprendido en el Capítulo 9

VARIOS

CLOAD

CSAVE

"almacenado"

"cargado"

** (indicador del cargado del programa)

CAPITULO 10

DECLARACIONES DE BUCLES O ITERACIONES DE PROGRAMACION

La principal diferencia entre una computadora y una calculadora es la habilidad de la computadora para hacer una misma cosa una y otra vez un número limitado de veces, a una velocidad muy alta.

La iteración FOR-NEXT es de gran importancia para el trabajo de la computadora y se usará muy a menudo, de ahora en adelante, en todas las áreas de la programación. Su simplicidad y flexibilidad son la razón de su efectividad.

Escribir y ejecutar el siguiente programa :

```
1Ø PRINT  "AUXILIO... MI COMPUTADORA SE HA VUELTO LOCA!"
2Ø GOTO 1Ø
```

Como puede apreciarse, la computadora está imprimiendo constantemente la línea 1Ø y continúa indefinidamente si no se le dice que pare, (oprimir la tecla [BREAK]).

Esto que hemos creado se llama "iteración sin fin". Como se recordará, en nuestros programas anteriores se usaba la declaración GOTO, para ejecutar un programa, introduciendo distintos datos de entrada. La línea 2Ø de nuestro programa, es una declaración GOTO incondicional, la cual es la causante de que la computadora vaya hacia adelante y hacia atrás, entre las líneas 1Ø y 2Ø contínuamente, si no se la detiene. Esta idea va a resultar de gran utilidad.

Vamos a modificar y ejecutar el programa de la siguiente forma :

```
 8  FOR N= 1 TO 5
1Ø  PRINT "AUXILIO... MI COMPUTADORA SE HA VUELTO LOCA!"
2Ø  NEXT N
3Ø  PRINT "NO... ESTA BAJO CONTROL".
```

Resultado y explicación :

La línea 1Ø aparece 5 veces en pantalla y a continuación se ha escrito la línea 3Ø. La iteración FOR NEXT de las líneas 8 y 2Ø ha hecho posible que la computadora formara un ciclo a través de las líneas 8, 1Ø y 2Ø 5 veces y después ha continuado con el resto del programa. Cada vez que la computadora llega a la línea 2Ø, encuentra "NEXT N", lo que permite que el valor de N se incremente en 1, saltando nuevamente a la declaración FOR N que fué la que inició la iteración. La declaración NEXT es incondicional, mientras N sea menor que 5 debido a que la línea 8 dice FOR N = 1 TO 5.

Después de haber pasado por quinta vez por la iteración, se rompe el ciclo y la ejecución del programa continúa. La declaración FOR-NEXT ha controlado de esta forma la iteración sin fin anterior.

LA FUNCION "STEP"

En ocasiones es conveniente incrementar el ciclo FOR-NEXT en algún valor diferente a uno. La función STEP lo permite. Cambiar la línea 8 a

```
8 FOR N = 1 TO 5 STEP 2
```

... y(RUN).

La línea 1Ø aparece en pantalla sólamente tres veces (cuando N = 1, N= 3 y N= 5). En la primera pasada (con N= 1) al llegar a NEXT N, se incrementa el valor de N en dos y así se obtiene N= 3 dando origen a la 2$^{da.}$ pasada; al llegar nuevamente a

NEXT N, se vuelve a incrementar en dos el valor de N pasando a valer 5 en la tercera pasada.

Las iteraciones FOR-NEXT, pueden ser incrementadas en cualquier número entero, incluso números negativos. El motivo por el cual pueda ser interesante el incremento de números negativos, se verá más adelante. Mientras tanto, cambiemos la línea 8 como sigue

```
8 FOR N = 5 TO 1 STEP -1
```

... y RUN.

Cinco pasadas del ciclo incrementando hacia abajo de 5 a 1 es exactamente lo mismo que incrementar hacia arriba de 1 a 5. La línea 1Ø se imprime también cinco veces.

MODIFICACIONES DE LA ITERACION FOR-NEXT

En el supuesto de que deseemos imprimir las líneas 1Ø y 3Ø cinco veces, en forma alternada, ¿cómo habremos de modificar el programa a que venimos haciendo referencia para lograrlo?

Se recomienda intentar hallar la solución cambiando de posición la línea NEXT N.

En efecto, si la línea 2Ø se mueve a la 4Ø se obtiene :

```
AUXILIO...MI COMPUTADORA SE HA VUELTO LOCA!
NO ... ESTA BAJO CONTROL.
AUXILIO...MI COMPUTADORA SE HA VUELTO LOCA!
NO...ESTA BAJO CONTROL.
```

... etc., tres veces más.

¿Cómo modificar el programa para que la línea 1Ø se imprima 5 veces y la 3Ø 3 veces?

El nuevo programa deberá ser :

```
 8  FOR N = 1 TO 5
1Ø  PRINT "AUXILIO...MI COMPUTADORA SE HA VUELTO LOCA!
2Ø  NEXT N
25  FOR M = 1 TO 3
3Ø  PRINT "NO... ESTA BAJO CONTROL".
4Ø  NEXT M
```

Este programa consta de dos iteraciones controladas. La primera efectúa una operación 5 veces y una vez finalizada, la segunda efectúa otra operación 3 veces. Puede observarse el uso de letras diferentes (N yM) para cada iteración. Sin embargo, se podía haber usado la misma debido a que las iteraciones son completamente independientes.

El uso de la declaración FOR-NEXT, no constituye una novedad fundamental en la filosofía de programación que se ha desarrollado hasta ahora, pero sí es una forma rápida y sencilla de ejecutar operaciones repetidas. Para demostrar esto último y de manera que sirva como repaso, se puede pensar en editar un programa que haga lo mismo que el del ejercicio anterior, pero sin usar la iteración FOR-NEXT.

```
 8  N = 1
1Ø  PRINT "AUXILIO MI COMPUTADORA SE HA VUELTO LOCA!"
15  N = N + 1
2Ø  IF N<6  THEN 1Ø
3Ø  PRINT "NO ESTA BAJO CONTROL"
4Ø  IF N<8  THEN 15
```

Explicación :

La línea 8 inicializa el valor de N, (en nuestro caso vale 1 pero podía haber tomado cualquier otro valor). Pasando por la línea 1Ø que imprime el texto entre comillas, se llega a la línea 15 que incrementa el valor de N en una unidad. La línea 2Ø, manda el programa a la línea 1Ø si el valor de N es menor que 6 con lo cual se establecen las 5 operaciones de escritura. Cuando N es igual a 6 se salta a la línea 3Ø que imprime el texto y se llega a la 4Ø que manda el programa a la 15 aumentando el valor de N en una unidad más. Esto permite realizar los 3 ciclos de escritura del segundo texto, ya que, cuando N alcanza el valor 8 el programa finaliza.

Hay una idea que sobresale del conjunto de lo explicado últimamente y es la posibilidad de programación diferente para un problema dado. El objetivo del buen programador consiste por tanto, en desarrollar el programa más rápido y sencillo para lo cual es fundamental acostumbrarse a los "buenos hábitos de programación", que se van reseñando a lo largo del manual.

También podemos realizar el programa decrementando el valor de N. Realizar los cambios oportunos en el programa residente para que la línea 15 contenga :

```
15  N = N - 1
```

SOLUCION :

```
 8 N = 8
2Ø IF N>3 THEN 1Ø
4Ø IF N>1 THEN 15
```

El siguiente ejemplo es una aplicación más de la iteración FOR-NEXT.

Supongamos que queremos imprimir una tabla que muestre cómo el tiempo que tarda en volar de Boston a San Diego, varía de acuerdo con la velocidad a la cual estamos viajando. Recordemos la fórmula D = V * T. Imprimamos el tiempo de vuelo requerido para cada velocidad entre 2ØØ KPH y 1ØØØ KPH con incrementos de 2ØØ KPH. El programa es el siguiente :

```
1Ø  REM * TABLA DE VUELO DEL TIEMPO EN FUNCION DE LA VELO-
    CIDAD *
2Ø  CLS
3Ø  D = 3ØØØ
4Ø  PRINT  "  BOSTON A SAN DIEGO "
5Ø  PRINT
6Ø  PRINT "VELOCIDAD (KPH)", "TIEMPO (HORAS)", "DISTANCIA
    (KILOMETROS)".
7Ø  PRINT
8Ø  FOR R = 2ØØ TO 1ØØØ STEP 1ØØ
9Ø  T = D/V
1ØØ PRINT  V,T,D,
11Ø NEXT V
```

Introducir el programa y ejecutarlo (RUN).

En realidad se está resolviendo el problema del capítulo 3 nueve veces seguidas para diferentes valores, e imprimiendo los resultados. La pantalla debe verse como sigue :

BOSTON A SAN DIEGO

VELOCIDAD (KPH)	TIEMPO (HORAS)	DISTANCIA (KMS)
2ØØ	15	3ØØØ
3ØØ	1Ø	3ØØØ
4ØØ	7.5	3ØØØ
5ØØ	6	3ØØØ
6ØØ	5	3ØØØ

VELOCIDAD (KPH)	TIEMPO (HORAS)	DISTANCIA (KMTROS.)
7ØØ	4.28571	3ØØØ
8ØØ	3.75	3ØØØ
9ØØ	3.33333	3ØØØ
1ØØØ	3	3ØØØ

Análisis del programa :

1. La declaración REM identifica el programa para usos posteriores.

2. La línea 2Ø usa la declaración CLS para borrar la pantalla y tener así un lugar limpio para escribir de una manera descendente. Puede ejecutarse el programa sin esta línea para ver lo que sucede.

3. La línea 3Ø inicializa el valor de D. D permanece con su valor inicial.

4. La línea 4Ø imprime el título de la tabla a doble espacio para que sea más legible.

5. Las líneas 5Ø y 7Ø usan PRINT vacíos para dejar espacios en la tabla.

6. La línea 6Ø imprime los títulos de las columnas de la tabla, y utiliza el espaciamiento automático de zonas para colocar esos títulos (la coma).

7. La línea 8Ø establece la iteración FOR-NEXT además de su incremento (STEP); ó dicho más claramente, inicializa la velocidad (V) en 2ØØ KPH, y pasa a través de la iteración con valores de V incrementados en 1ØØ KPH hasta llegar al valor final de 1ØØØ KPH. La línea 11Ø constituye la otra mitad de la iteración.

8. La línea 9Ø contiene la fórmula que calcula la respuesta.

9. La línea 1ØØ imprime los tres valores. Están localizados bajo sus títulos por el espaciamiento automático de zonas (las comas).

10. Las líneas 9Ø y 1ØØ deben estar un poco más espaciadas del resto del texto del programa.
Esta es una técnica de programación característica de una iteración (cíclica), la cual hace que la lectura y la búsqueda de errores sean más sencillas. Se utilizarán con bastante frecuencia a medida que avancemos. Hay que tratar de adoptar buenas prácticas de programación como ésta al hacer los ejercicios. Este espaciamiento ocupa un poco más de sitio en la memoria, y en programas largos al llegar a su forma final muchas veces se omite.

Se recomienda almacenar este programa en la cinta de la casette, porque se usará en el capítulo siguiente.

NOTA :

CLS en un programa efectúa la misma función que la tecla [CLEAR] del teclado. (Pero no se puede utilizar la tecla [CLEAR] como parte de un programa).

Las iteraciones encadenadas que se han venido explicando reciben el nombre de iteraciones-DO.

Material Aprendido en el Capítulo 10

INSTRUCCIONES	DECLARACIONES	VARIOS
	FOR-NEXT	Incremento
	CLS	Disminución
	STEP	Inicializa
		Tecla BREAK
		Tecla CLEAR
		Presentación en forma "descendente"
		Presentación en forma "de lista"
		"ITERACION-DO"

CAPITULO 11

CONTROL EN LA EJECUCION DE PROGRAMAS

Carguemos el programa que se grabó en el capítulo 10. Vamos a modificarlo y a ejecutarlo de modo que la velocidad y el tiempo se calculen e impriman cada 5Ø KPH en lugar de cada 1ØØ KPH correspondientes al programa inicial.
Es decir :

```
8Ø  FOR R = 2ØØ TO 1ØØØ STEP 5Ø
```

Resultado : La impresión es tan rápida que nos es imposible leerla, y para cuando se detiene, la parte de arriba ya no aparece en pantalla.

Existen varias soluciones :

1. Si se oprime cualquiera de las teclas, se detendrá la ejecución del programa. Ejecutemos varias veces el programa oprimiendo diferentes teclas (y la barra de espacios), para observar qué sucede.
   Ejecutémoslo una vez más utilizando esta vez la tecla ↑ (flecha ascendente), para paralizar la emisión en pantalla. Con esta tecla, se detiene y reinicia limpiamente. Esta es la que se debe utilizar para detener la emisión en pantalla temporalmente.

2. También se puede colocar "una pausa" dentro del programa. La pantalla se llenará, se detendrá un momento y automáticamente proseguirá.

NOTAS :

- Como puede verse, el hecho de oprimir una tecla no sólo detiene la ejecución sino que además introduce el propio número o letra.

- Existe otra tecla que se puede oprimir pero no es muy útil. La tecla BREAK , que es aún más desordenada. Para reiniciar después de oprimir BREAK se oprime RUN o CONT.

ITERACION CRONOMETRADA

Para entender qué es una iteración cronometrada vamos a dejar el programa " de tiempo de vuelo" en la computadora, y a introducir un segundo programa.

Escribir y ejecutar :

```
9  END
1  PRINT  "NO SE VAYAN"
2  FOR X = 1 TO 5ØØØ
3  NEXT X
5  PRINT  "PROGRAMA CRONOMETRO TERMINADO".
```

La línea 9 se coloca para separar los programas.
Si se ha efectuado todo correctamente, la línea 5 debe haber tardado en aparecer, unos 10 segundos. Como puede observarse, este programa se basa en el tiempo que tarda el ordenador en efectuar iteraciones FOR-NEXT, (aproximadamente 1 segundo cada 5ØØ).

Cambiar el programa para crear un retardo de 30 segundos y compararlo con el que nos dé el reloj.

Respuesta : 2 FOR X= 1 TO 15ØØØ

EJERCICIO 11-1 : Usando el espacio que existe entre las líneas 1 a 8, diseñar un programa que pregunte cuantos segundos de retardo se desean, que permita introducir un número y que luego ejecute el retardo y al terminarlo imprima cuántos segundos ha tardado. La respuesta se encuentra al final del capítulo.

MANEJO DEL LISTADO DE PROGRAMAS LARGOS

Cuando se efectúa un listado en problemas largos, (como los dos que tenemos en nuestra computadora), aparece el problema de que no se puede ver todo el programa.
La solución más sencilla en nuestro caso, para ver el resto del programa, está en pulsar sucesivamente la tecla (↑); cada vez que se oprime esta tecla, el listado asciende una línea y por tanto, aparece una nueva en la parte inferior de la pantalla.

Otra solución es usar una versión de LIST un poco diferente. LIST # # # , donde # # # representa la línea de comienzo de listado.

Escribamos para nuestro programa :

LIST 5Ø

La computadora da un listado que empieza con la línea 5Ø y llena la pantalla con 16 líneas o si el programa es corto, imprime hasta el final.

LIST # # # y la ↑ (flecha ascendente), pueden ser usadas para localizar cualquier parte que se desee de un programa muy largo.
Una vez más, debe estar presente el apuntador si se desea continuar o hacer cualquier otra cosa.
Aparte del hecho de utilizar una (o ambas) de las técnicas anteriores para obtener el final del listado (LIST) y encontrar el apuntador, existe una manera más rápida, una vez que se haya encontrado lo que se deseaba en el listado. Basta, para ello, con pulsar la tecla [ENTER] una o dos veces.

EJECUCION DE UN PROGRAMA EN PRESENCIA DE VARIOS

Cuando se tienen en la computadora dos o más programas seguidos (como en el caso de este capítulo), es posible la ejecución de cualquiera de ellos, en el momento que se desee. En nuestro ejemplo, bastaría con teclear RUN para ejecutar el primero. En cambio si se quiere ejecutar el segundo, hay que teclear RUN # # #, incluyendo en " # # #" el número de línea de comienzo del programa (en nuestro caso 1Ø). Como se podía esperar, el procedimiento es similar al del caso anterior del listado LIST # # # .

EJECUCION CONTROLADA DEL PROGRAMA

Vamos a introducir ahora una iteración controlada, dentro de nuestro programa de los vuelos Boston-San Diego. Para ello, primero borraremos las líneas 2,3,4,5,6,7 y 9, escribiendo cada uno de los números, seguidos por un [ENTER].

Una manera de detener el paso rápido de la información en nuestra tabla, es introducir un STOP. Escribamos :

```
85 IF R = 6ØØ STOP
... y RUN
```

Sabemos que R va a alcanzar el valor 6ØØ hacia la mitad de la tabla. Observemos cómo la tabla se detiene en 6ØØ, descendiendo a la línea 85. La pantalla debe mostrar la primera parte de la tabla y [BREAK] AT 85 .

Esto indica que el programa se ha detenido en la línea 85. Para que el programa continúe, se teclea :

```
CONT
```

y automáticamente se imprime el resto de la tabla.

El siguiente paso que vamos a dar, es construir un retardo dentro del programa para que no haya necesidad de detener totalmente la ejecución del mismo. Esto se consigue escribiendo :

```
85  IF R<>6ØØ THEN 9Ø
87  FOR X = 1 TO 5ØØ
88  NEXT X
```

... y RUN

Mientras R no sea igual a 6ØØ el programa se salta la iteración de retardo de las líneas 87 y 88; pero cuando alcanza este valor, el programa pasa a la línea siguiente (87) y la iteración retarda la ejecución en un segundo aproximadamente.

EJERCICIO 11-2 : Modificar el programa residente de manera que (KPH) aparezca debajo de VELOCIDAD, (HORAS) aparezca debajo de TIEMPO y (KILOMETROS) aparezca debajo de DISTANCIA.

EJERCICIO 11-3 : Diseñar, escribir y ejecutar un programa que calcule e imprima los ingresos anuales, mensuales, semanales y diarios basados en una semana de cuarenta horas, un mes de 1/12 de año y un año de 52 semanas. Hacer esto, para ingresos entre $5ØØØ y $25ØØØ en incrementos de $1ØØØ. Documentar el programa con declaraciones REM cuando sea necesario para explicar las ecuaciones que se vayan a crear.

EJERCICIO 11-4 : Diseñar, escribir y ejecutar un programa que diga cuántos días se tendrá que trabajar, empezando por una peseta al día, si el salario se duplica cada día, de manera que se sepa qué día se ganará por lo menos un millón de pesetas.

Inclúyanse columnas que muestren el día (su número), el salario diario, y el ingreso total en la fecha. Hacer que el programa se detenga después de imprimir el día que el salario diario sea de un millón de pesetas ó más.

Una de las características de la computadora es que permite llegar a la solución de un problema por el método del "tanteo", debido a su gran rapidez y agilidad de cálculo. Los siguientes ejercicios constituyen una viva muestra de lo que acabamos de decir.

EJERCICIO 11-5 : Se tiene un rollo de mil metros de largo de alambre y se quiere cercar con él un pastizal rectangular.

Usando todo el alambre, determinar qué longitud y anchura permitirán encerrar el máximo número de metros cuadrados. Dejar que la computadora intente varios valores para L y W e imprima el área cercada para cada par de valores L y W.

La fórmula para el Area = Longitud por Ancho

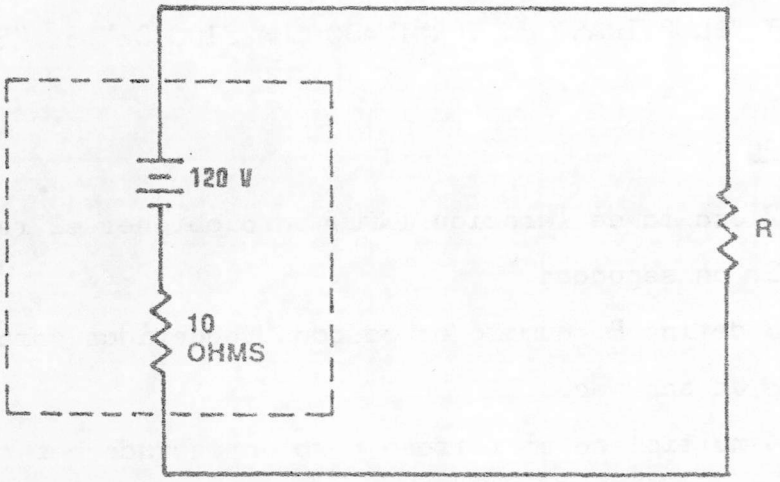
ó A = L * W

EJERCICIO 11-6 : Con los valores dados en la figura diseñar, escribir y ejecutar un programa que tome valores de resistencia que van desde 1 a 2Ø ohms, en incrementos de 1 ohm, e imprima las respuestas con los datos siguientes :

1. Valor de la resistencia de carga (de 1 a 2Ø ohms).
2. La potencia perdida en la fuente (el cuadrado de la corriente del circuito, multiplicado por la resistencia de la fuente) I^2 * 1Ø .
3. La potencia suministrada a la carga (el cuadrado de la corriente del circuito multiplicada por la resistencia de carga) I^2 * R .

NOTA :

La corriente del circuito se encuentra dividiendo el voltaje de la fuente (12Ø voltios) por la resistencia total del circuito (resistencia de carga + 1Ø ohms de resistencia de la fuente). Todo está de acuerdo con la ley de Ohm, V = I * R), y la ley de WATT, P = I * V).



Material Aprendido en el Caítulo 11

ORDENES	DECLARACIONES	VARIOS
LIST # # #	STOP	ITERACION CRONOMETRADA
RUN # # #		Flecha ascendente ↑
CONT		Método de tanteo

SOLUCION A LOS EJERCICIOS PROPUESTOS EN EL CAPITULO 11

RESPUESTA TIPO PARA EL EJERCICO 11-1 :

```
2 INPUT  "CUANTOS SEGUNDOS DE RETRASO DESEA?"; S
3 P = 5ØØ
4 D = S * P
5 FOR X = 1 TO D
6 NEXT X
7 PRINT "EL RETRASO HA TERMINADO. HA SIDO DE";S; "SEGUNDOS".
```

Explicación :

La línea 2 usa la declaración INPUT para obtener el retraso deseado, S, en segundos.
La línea 3 define P, número de pasadas requeridas para un retraso de un segundo.
La línea 4 multiplica el retraso para un segundo por el número de segundos deseados, y le llama a ese producto D.
La línea 5 inicia la iteración FOR-NEXT de 1 a lo que se requiera.
La línea 6 es la otra mitad de la iteración.
La línea 7 indica que el retraso se ha acabado, e imprime S, el número de segundos.

RESPUESTA TIPO PARA EL EJERCICIO 11-2 :

```
6Ø  PRINT  "VELOCIDAD", "TIEMPO", "DISTANCIA".
65  PRINT  " (KPH)", "(HORAS)",  "(KILOMETROS)"
```

Si se han tenido problemas en la resolución de este ejercicio, deben repasarse los capítulos anteriores.

RESPUESTA TIPO PARA EL EJERCICIO 11-3 :

```
  5  CLS
 1Ø  PRINT  "     ***      TABLA DE SALARIOS        *** "
 2Ø  PRINT
 3Ø  PRINT  " AÑO ","MES", "SEMANA", "DIA"
 4Ø  PRINT
 5Ø  FOR Y = 5ØØØ TO 25ØØØ STEP 1ØØØ
 55  REM* CONVIERTA EL INGRESO ANUAL EN MENSUAL*
 6Ø  M = Y/12
 65  REM *CONVIERTA EL INGRESO ANUAL EN SEMANAL*
 7Ø  W = Y/52
 75  REM *CONVIERTA EL INGRESO SEMANAL EN DIARIO *
 8Ø  D = W/5
1ØØ  PRINT  Y,M,W,D
11Ø  NEXT Y
```

NOTA :

La función FOR-NEXT-STEP está limitada a los números entre -32767 y +32767 (inclusive). Si intentamos que la máquina trabaje fuera de estos márgenes, obtendremos el mensaje HOW?

También se produce otro error menos obvio si todos sus números están dentro de la gama pero la suma del límite superior y la del tamaño del paso supera 32767. Por ejemplo, si se intenta ejecutar :

```
5Ø  FOR Y = 5ØØØ TO 327ØØ STEP 1ØØØ
```

en el Programa de Tabla de Salarios.

La manera de salir de este problema es usar límites superior e inferior y tamaño de paso más pequeños, y luego usar el factor de escala en la iteración para obtener el número más grande. Por ejemplo :

```
 5Ø  FOR Z = 5ØØ TO 327Ø STEP 1ØØ
 53  Y = Z ^ 1Ø
11Ø  NEXT Z
```

EJECUCION TIPO PARA EL EJERCICIO 11-3 :

TABLA DE SALARIOS

AÑO	MES	SEMANA	DIA
5ØØØ	416.667	96.1538	19.2308
6ØØØ	5ØØ	115.385	23.Ø769
7ØØØ	583.3333	134.615	26.9231

ETC.

RESPUESTA TIPO PARA EL EJERCICIO 11-4 :

```
 1Ø  R = 1
 2Ø  D = 1
 3Ø  T = 1
 35  CLS
 4Ø  PRINT  "DIA", "DIARIO", "TOTAL"
 5Ø  PRINT  "#  ", "RAZON",  "GANADO"
 6Ø  PRINT
 7Ø  PIRNT  D,R,T,
 8Ø  IF R > 1E6  END
 9Ø  R = R * 2
1ØØ  D = D + 1
11Ø  T = T + R
12Ø  GOTO 7Ø
```

EJECUCION TIPO PARA EL EJERCICIO 11-4 :

DIA #	DIARIO RAZON	TOTAL GANADO
1	1.ØØØØØØE-Ø2	1.ØØØØØØE-Ø2

DIA #	DIARIO RAZON	TOTAL GANADO
2	2E-Ø2	3E-Ø2
3	4E-Ø2	7E-Ø2
4	8E-Ø2	.15
5	.16	.31
6	.32	.63

ETC.

RESPUESTA TIPO PARA EL EJERCICIO 11-5 :

```
 1 REM * OBTENCION DEL AREA MAXIMA *
 5 CLS
1Ø PRINT  "CERCA DE ALAMBRE", "LONGITUD", "ANCHURA", "AREA"
2Ø PRINT  "(METROS)", "(METROS)", "(METROS)", "(METROS CUAD.)"
3Ø F = 1ØØØ
4Ø FOR L = Ø TO 5ØØ STEP 5Ø
5Ø W =  (F-2 * L)/2
6Ø A = L* W
7Ø PRINT  F,L,W,A
8Ø NEXT L
9Ø END
```

EJECUCION TIPO PARA EL EJERCICIO 11-5 :

CERCA DE ALAMBRE (METROS)	LARGURA (METROS)	ANCHURA (METROS)	AREA (METROS CUAD.)
1ØØØ	Ø	5ØØ	Ø
1ØØØ	5Ø	45Ø	225ØØ
1ØØØ	1ØØ	4ØØ	4ØØØØ
1ØØØ	15Ø	35Ø	525ØØ
1ØØØ	2ØØ	3ØØ	6ØØØØ
1ØØØ	25Ø	25Ø	625ØØ
1ØØØ	3ØØ	2ØØ	6ØØØØ

ETC.

RESPUESTA TIPO AL EJERCICIO 11-5 :

```
 5  CLS
 9  REM * FIJA EL AREA MAXIMA EN CERO *
1Ø  M = Ø
14  REM * FIJA LA LONGITUD DESEADA EN CERO *
15  N = Ø
19  REM *F ES EL TOTAL DE METROS DISPONIBLES *
2Ø  F = 1ØØØ
24  REM * L ES LONGITUD DE UN LADO DEL RECTANGULO *
25  FOR L = Ø TO 5ØØ STEP 5Ø
29  REM * L ES LA ANCHURA DE UN LADO DEL RECTANGULO *
3Ø  W =  (F - 2 * L)/ 2
35  A =  W* L
39  REM  *COMPARA A CON EL MAXIMO ACTUAL. REEMPLAZA SI ES
    NECESARIO *
4Ø  IF A< = M THEN GOTO 55
45  M = A
49  REM * TAMBIEN ACTUALIZA LA LONGITUD DESEADA ACTUAL*
5Ø  N = L
55  NEXT L
6Ø  PRINT  "DIMENSIONES CORRESPONDIENTES AL AREA MAXIMA"
65  PRINT N ; "METROS POR"; 5ØØ - N; "METROS PARA UN
    AREA TOTAL DE"; M; "METROS CUAD."
```

EJECUCION TIPO PARA EL EJERCICIO 11-5 :

```
   DIMENSIONES CORRESPONDIENTES AL AREA MAXIMA
25Ø METROS POR 25Ø METROS PARA UN AREA TOTAL DE 625ØØ METROS
CUAD.
```

RESPUESTA TIPO PARA EL EJERCICIO OPCIONAL 11-6 :

```
1Ø  REM * OBTENCION DE LA PAREJA OPTIMA RESISTENCIA INTERNA-
    RESISTENCIA DE CARGA *
2Ø  CLS
3Ø  PRINT  "RESISTENCIA", "POTENCIA DEL", "POTENCIA DE LA",
    "POTENCIA DE LA"
4Ø  PRINT  "DE CARGA", "CIRCUITO", "FUENTE", "CARGA"
5Ø  PRINT  "(OHMIOS)"
6Ø  PRINT
7Ø  FOR R = 1 TO 2Ø
8Ø  I = 12Ø/(1Ø + R)
9Ø  C = I * I * (1Ø + R)
1ØØ  S = I * I * 1Ø
11Ø  L = I * I * R
12Ø  PRINT R,C,S,L
13Ø  NEXT R
```

EJECUCION TIPO PARA EL EJERCICIO 11-6 :

RESISTENCIA DE CARGA (OHMIOS)	POTENCIA DEL CIRCUITO (VATIOS)	POTENCIA DE LA FUENTE (VATIOS)	POTENCIA DE LA CARGA (VATIOS)
1	13Ø9.Ø9	119Ø.Ø8	119.ØØ8
2	12ØØ	1ØØØ	2ØØ
3	11Ø7.69	852.Ø71	255.621

ETC.

CAPITULO 12

LA DECLARACION TAB

Ya conocemos tres formas de escribir el formato de salida PRINT.

Podemos :

1. Encerrar lo que queremos decir entre comillas, incluyendo espacios en blanco si son necesarios.
2. Separar los argumentos de la declaración PRINT con puntos y comas (;) a fin de imprimirlos muy juntos sobre la misma línea.
3. Separar los argumentos de la declaración PRINT con comas (,) para imprimirlos sobre la misma línea en cuatro zonas de impresión distintas.

Una cuarta forma es usando la función TAB, que es similar al espaciador TAB de una máquina de escribir ordinaria. Es de especial utilidad cuando la impresión consta de columnas de números con títulos. Escribir el siguiente programa y ejecutarlo.

```
1Ø PRINT TAB (5); "EL"; TAB (2Ø); "TOTAL"; TAB(35); "GASTADO"
2Ø PRINT TAB (5); "PRESUPUESTO"; TAB(2Ø); "AÑO"; TAB(35);
   "ESTE"
3Ø PRINT TAB (5); "CATEGORIA"; TAB(2Ø); "PRESUPUESTO";
   TAB(35); "MES"
```

La pantalla debe mostrar :

```
EL            TOTAL          GASTADO
PRESUPUESTO   ANUAL          ESTE
CATEGORIA     PRESUPUESTO    MES
```

Como puede apreciarse, el número entre paréntesis de la declaración TAB, define la posición del primer carácter del texto partiendo del margen izquierdo de la pantalla.

EJERCICIO 12-1 : Escribir un programa usando las tres formas de PRINT que se muestran a continuación :

1. PRINT " ", " "," "
2. PRINT " "
3. PRINT TAB ( ); " "; TAB (); " ";TAB( );" "

Colocar los títulos dados en el ejemplo anterior, usando la forma 1 para la primera línea, la forma 2 para la segunda línea y la forma 3 (la forma TAB) para la tercera línea. Hay que fijarse en que ya que la forma 1 utiliza el formato automático, las demás deben adecuarse a este formato.

La forma TAB, tiene algunas peculiaridades que conviene destacar. El punto y coma (;) después de TAB( ) puede ser sustituído por la coma (,) sin que por ello cambie la ejecución de la declaración PRINT. También hay que tener en cuenta, que cuando se imprimen números, la computadora inserta automáticamente un espacio para permitir la colocación del signo - ó + .

Teclear y ejecutar el siguiente programa :

```
1Ø  A = 3
2Ø  B = 5
3Ø  C = A + B
4Ø  PRINT TAB(1Ø);"A"; TAB(2Ø); "B"; TAB(3Ø); "C"
5Ø  PRINT TAB(1Ø); A; TAB(2Ø); B; TAB(3Ø); C
```

La pantalla debe mostrar

```
A          B          C
 3          5          8
```

Observar el desplazamiento de los números de un espacio, con respecto a las letras.
Tendrá que tenerse en cuenta cuando se quieran alinear títulos y respuestas.

Cambiar la línea 2Ø para que diga

```
2Ø  B = -5
```

... y RUN. Observar por qué es necesaria la alineación.

DIVISION DE LAS LINEAS LARGAS

¿Qué pasaría si se quisieran imprimir un gran número de títulos o de respuestas sobre la misma línea, pero no se tuviera el espacio suficiente sobre la línea del programa para poder poner todas las declaraciones TAB ?
Resulta sencillo averiguarlo con el siguiente ejemplo.
Teclear y ejecutar :

```
1Ø A = 1
2Ø B = 2
3Ø C = 3
4Ø D = 4
5Ø E = 5
6Ø F = 6
7Ø G = 7
8Ø H = 8
9Ø I = 9
1ØØ J = 1Ø
2ØØ PRINT "A"; TAB(5Ø); "B"; TAB(1Ø);"C"; TAB(15); "D";
21Ø PRINT TAB(2Ø); "E"; TAB(25);"F"; TAB(3Ø);"G";
22Ø PRINT TAB(35);"H"; TAB(4Ø);"I"; TAB(45);"J"
3ØØ PRINT A; TAB(5); B; TAB(1Ø);C; TAB(15);D; TAB(2Ø);
31Ø PRINT E; TAB(25); F; TAB(3Ø);G; TAB(35);H; TAB(4Ø);
32Ø PRINT I; TAB(45); J
```

El punto y coma (;) que aparece después de cada instrucción ha resuelto nuestro problema. Hace que el final de una línea de impresión (PRINT) continúe hacia la siguiente —línea de impresión (PRINT) sin activar el retorno del carro. La combinación de TAB y los puntos y comas (;) al final permite un número infinito de posibilidades y mucha flexibilidad para diseñar el formato de salida.

LINEAS DE DECLARACIONES MULTIPLES

El TRS-80 permite que las declaraciones se puedan encadenar, dentro de la misma línea, simplemente separándolas con dos puntos.
Reemplazar las líneas 1Ø a 1ØØ con lo que sigue, y ejecutar de nuevo el programa :

```
1Ø  A =1 : B =2 : C =3 : D =4 : E=5 : F =6 : G =7 :
    H =8 : I = 9 : J = 1Ø
```

La computadora lee de izquierda a derecha como si cada declaración fuera un número de línea diferente.

La única excepción a esta regla se encuentra en la declaración IF -THEN con la que existen consideraciones especiales. Cuando se utilizan declaraciones múltiples, hay que tener en cuenta que no se puede mandar al programa que vaya a una declaración intermedia de la línea de declaraciones múltiples. La ventaja de este tipo de líneas está en el ahorro de espacios, haciendo más cortos los programas.

EJERCICIO 12-2 : Modificar la solución dada al ejercicio 11-3 para que aparezca en la pantalla el importe cobrado por hora de trabajo. Usar la función TAB para hacer que la tabla muestre las 5 columnas.

EJERCICIO 12-3 : (Opcional) Modificar la solución dada al problema especial 11-6 usando la función TAB de manera que la impresión incluya la resistencia interna en la quinta columna.

Material Aprendido en el Capítulo 12

VARIANTES DE IMPRESION	VARIOS
TAB	Punto y coma finales
	Líneas de declaraciones múltiples

SOLUCION A LOS EJERCICIOS PROPUESTOS EN EL CAPITULO 12

RESPUESTA TIPO PARA EL EJERCICIO 12- 1 :

```
10  PRINT  "EL", "TOTAL", "GASTADO"
20  PRINT  "PRESUPUESTO    ANUAL            ESTE"
30  PRINT TAB(0); "CATEGORIA"; TAB(16); "PRESUPUESTO";
    TAB(32); "MES"
```

RESPUESTA TIPO PARA EL EJERCICIO 12-2 :

```
30  PRINT TAB(1); "AÑO"; TAB(12); "MES"; TAB(25);"SEMANA";
40  PRINT TAB(38); "DIA"; TAB(51); "HORA"
85  REM  *CONVERSION DEL INGRESO SEMANAL EN INGRESO POR
    HORA *
90  H * W/40
100 PRINT TAB(1); Y; TAB(12); M; TAB(25); W; TAB(38);D;
    TAB(51);H
```

EJECUCION TIPO PARA EL EJERCICIO 12-2 :

TABLA DE SALARIOS

AÑO	MES	SEMANA	DIA	HORA
5000	416.667	96.1538	19.2308	2.40385
6000	500	115.385	23.0769	2.88462

ETC.

RESPUESTA TIPO PARA EL EJERCICIO 12-3 :

```
30  PRINT "RESIST"; TAB(10); "RESIST"; TAB(21); "POTENCIA";
35  PRINT TAB(36); "POTENCIA"; TAB(51); "POTENCIA"
40  PRINT "INTER"; TAB(10); "CARGA"; TAB(21); "CIRCUITO";
45  PRINT TAB(36); "FUENTE"; TAB(51); "CARGA"
```

```
5Ø  PRINT "(OHMIOS)"; TAB(1Ø);"(OHMIOS)"; TAB(21);
    "(VATIOS)";
55  PRINT TAB(36); "(VATIOS)"; TAB(51);"(VATIOS)"

12Ø PRINT "     1Ø"; TAB(1Ø); R; TAB(2Ø); C; TAB(35);
    S; TAB(5Ø), L
```

EJECUCION TIPO PARA EL EJERCICIO 12-3 :

RESIST INTER (OHMIOS)	RESIST CARGA (OHMIOS)	POTENCIA CIRCUITO (VATIOS)	POTENCIA FUENTE (VATIOS)	POTENCIA CARGA (VATIOS)
1Ø	1	13Ø9.Ø9	119Ø.Ø8	119.ØØ8
1Ø	2	12ØØ	1ØØØ	2ØØ
1Ø	3	11Ø7.69	852.Ø71	255.621

ETC.

CAPITULO 13

ITERACIONES ANIDADAS

Escribir y ejecutar el programa :

```
1ØFOR A = 1 TO 3
2Ø PRINT "A LOOP"
3Ø  FOR B = 1 TO 2
4Ø   PRINT "    ", "B LOOP"
5Ø  NEXT B
6ØNEXT A
```

El resultado debe ser :

```
A LOOP
                    B LOOP
                    B LOOP
A LOOP
                    B LOOP
                    B LOOP
A LOOP
                    B LOOP
                    B LOOP
```

La Pantalla muestra el funcionamiento de la iteración FOR-NEXT "anidada". "Anidamiento" es un término usado en el mismo sentido en que se usa cuando se dice que los vasos para beber agua están "anidados" cuando se guardan uno dentro del otro para ahorrar espacio.
Algunos tipos de sillas portátiles, cajas de cartón vacías, etc... pueden ser "anidadas". Caben una dentro de la otra para su fácil apilamiento.

Analicemos el programa línea a línea :

La línea 1Ø establece la primera iteración FOR-NEXT, llamada A y ordena que sea ejecutada 3 veces.

La línea 2Ø imprime "A LOOP" para que sepamos de dónde sale el programa. Obsérvese cómo esta línea del programa se ha espaciado varios lugares para recalcar que se encuentra anidada dentro de la iteración "A".

La línea 3Ø establece la segunda iteración, llamada B, y ordena que sea ejecutada dos veces. Está aun más espaciada de manera que pueda instantáneamente verse que está "encerrada" aún más profundamente dentro de la iteración "A".

La línea 4Ø imprime dos cosas : Primero el espacio que se muestra entre las dos comillas, luego la coma nos lleva dentro de la siguiente zona de impresión donde se imprime la iteración "B".
Esto hace posible que se distingan claramente sobre la pantalla la iteración A y la iteración B.

La línea 5Ø completa la iteración "B" y devuelve el control a la línea 3Ø para que se realicen tantas iteraciones como indique la línea 3Ø. Hasta aquí, hemos impreso una "A" y una "B".

La línea 6Ø finaliza la primera pasada a través de la iteración "A", y manda el control de nuevo a la línea 1Ø, inicio de la iteración A. La iteración "A" se realiza 3 veces y la "B", 6 veces (3 por 2).

Para entender mejor el sentido de la iteración anidada, modifiquemos la línea 1Ø, así :

```
1Ø FOR A = 1 TO 5
```

En la ejecución, puede apreciarse que la A se ha impreso 5 veces y por tanto la B 1Ø veces.

Ahora se puede cambiar la línea 3Ø para que diga :

```
3Ø FOR B = 1 TO 4
```

RUN. La A se ha impreso 5 veces y la B 20. Recordar que se puede parar la emisión de datos apretando las teclas (↑), (↓), (→) ó pulsando BREAK (teclear CONT para continuar).

Hay que poner especial atención cuando se trabaja con iteraciones anidadas para no cometer errores.

Si se cambian las líneas :

```
5Ø   NEXT A
6ØNEXT B
```

... y RUN

La computadora dice :

```
WHAT?
6Ø NEXT B?
```

El error se debe a que la iteración B, no se encuentra anidada dentro de la iteración A. Tenemos la parte FOR de la iteración B dentro de la iteración A, pero la parte NEXT está fuera de ella. En un capítulo posterior se verá lo que se llama "diagramas de flujo", que son una forma de poder planear nuestros programas y evitar este tipo de problemas.

COMO SALIR DE LAS ITERACIONES

Para salir de las iteraciones cuando se desea, se pueden intercalar en el programa declaraciones IF-THEN GOTO. Agregar estas líneas en el programa anterior :

```
5Ø NEXT B
55 IF A = 2 GOTO 1ØØ
6Ø NEXT A
```

```
99 END
1ØØ PRINT "A FUE IGUAL A 2. FIN DE EJECUCION"
```

... y RUN

Como muestra la pantalla, se ha salido de la iteración A, cuando A se iguala a 2 y se llega a la línea de prueba en 55. El END de la línea 99 es sólamente un obstáculo que se pone como precaución para detener la computadora en caso de error y que no pase por la línea 100, a menos que se le ordene que lo haga. Normalmente estos"bloques de protección", no son necesarios en programas tan sencillos, pero deben usarse para proteger declaraciones específicas (GOTO, IF-THEN), contra posibles errores.

Un simil que visualiza muy bien el concepto de iteraciones anidadas puede ser el de las vasijas que muestra la "FIGURA" :

FIGURA

Supongamos que un grifo suministra agua a la vasija C. Esta vasija, se va llenando, y cuando se completa, vertemos su contenido en la vasija B. Habrá que repetir esta operación un determinado número de veces para que la vasija B se llene y cuando esto suceda, vertemos su contenido en la vasija A. Naturalmente, hay que volver a empezar a llenar la vasija C ya que tanto la B como la C se encuentran ahora vacías; sólo la A tiene una cierta cantidad de agua. Repitiendo las operaciones de llenado de C y B, se termina por llenar la vasija A.

EJERCICIO 13-1 : Teclear el programa original que se encuentra al inicio de este capítulo. Contiene una iteración B anidada dentro de una iteración A. Hacer los cambios necesarios en dicho programa de manera que una nueva iteración llamada C quede anidada dentro de la iteración B, y que imprima "C LOOP" cuatro veces por cada pasada por la iteración B.

EJERCICIO 13-2 : Modificar el programa residente de manera que sea el mismo programa que se encuentra en la respuesta al ejercicio 13-1.

Hacer las adiciones necesarias a este programa de manera que una nueva iteración llamada "D" quede anidada dentro de una iteración C, y que imprima "D LOOP" cinco veces por cada pasada por la iteración C.

Material Aprendido en el Capítulo 13

VARIOS

Iteraciones FOR-NEXT ANIDADAS

Bloques de protección END

SOLUCION A LOS EJERCICIOS PROPUESTOS EN EL CAPITULO 13

RESPUESTA TIPO PARA EL EJERCICIO 13-1 :

```
1Ø FOR A = 1 TO 3
2Ø PRINT "ITERACION A"
3Ø FOR B = 1 TO 2
4Ø PRINT "  ", "ITERACION B"
42       FOR C = 1 TO 4
44        PRINT "  "," ", "ITERACION C"
48       NEXT C
5Ø      NEXT B
6ØNEXT A
```

RESPUESTA TIPO PARA EL EJERCICIO 13-2 :

Este programa es igual a la respuesta del ejercicio 13-1 con las siguientes adiciones :

```
45       FOR D = 1 TO 5
46        PRINT "  ", " ", "  ", "ITERACION D"
47       NEXT D
```

NOTA :

Para entender mejor cómo actúan las iteraciones anidadas, se puede detener varias veces la ejecución del programa.

CAPITULO 14

LA FUNCION INTEGER

La función INTEGER, INT(X) nos permite "redondear" cualquier número grande o pequeño, positivo o negativo, a un número entero.

Teclear NEW para borrar cualquier programa viejo, luego escríbase :

```
30 X = 3.14159
40 Y = INT(X)
50 PRINT "Y = "; Y
```

... y RUN.

En la pantalla debe aparecer

```
Y = 3
```

El resultado muestra el redondeo del número π a su valor entero más bajo.
Si se cambia la línea 30 por :

```
30 X = -3.14159
```

... y RUN. La respuesta se redondea al valor entero - 4.

```
Y = -4
```

¿Qué clase de redondeo es este? Sencillo. La función INT SIEMPRE REDONDEA al número entero siguiente MAS BAJO.
Hace a un número positivo menos positivo, y a un número negativo más negativo (o lo que es igual menos positivo).

NOTA :

EL BASIC NIVEL I permite a la función INT(X) trabajar sólamente con números mayores que -32767 y menores que +32767.
EL BASIC NIVEL 2 no tiene esta restricción.

El uso de un valor de X fuera de este rango, hace que la computadora se detenga y pregunte HOW?

CONTROL DE LA PRECISION POR MEDIO DE LA FUNCION INTEGER

Escribir y ejecutar :

```
30 X = 3.14159
40 Y = INT(X)
50 Z = X - Y
60 PRINT " X ="; X
70 PRINT "Y = "; Y
80 PRINT "Z = "; Z
```

... y RUN.

```
X = 3.14159
Y = 3
Z = .141589
```

Hemos partido el valor de X en su valor entero, al cual lo hemos llamado Y, y su valor decimal al que hemos llamado Z.

Las líneas 60, 70 y 80 simplemente imprimen los resultados.

Ahora bien, si nos fijamos existe una diferencia entre Z y la parte decimal de X ¿A qué se debe?

No tiene nada que ver con la función INT. En el capítulo 8 se habló acerca de la precisión de la computadora (siempre hay que observar la precisión del último lugar o de los dos últimos lugares decimales). Los usuarios de la TRS-80 que posean el BASIC NIVEL 2 no observarán esta rutina de "redondeo de error".

Existe una forma de controlar la precisión de los resultados en el BASIC NIVEL 1.

Tiene que ver con el redondeo artificial de la fracción al número de lugares decimales deseados, forzando la computadora para que imprima sólo aquellos dígitos que estén "apropiadamente redondeados".

Ejemplo : Supongamos que sólo se necesita un π con 3 decimales. Escribir y ejecutar el siguiente programa tecleando primeramente NEW.

```
10 X = 3.14159
20 X = X + .0005
30 X = INT(X * 1000)/1000
40 PRINT X
```

Se pueden dar otros valores a X, (asegurándose de que X * 1000, no sea demasiado grande para la función INT).

Al agregar .0005 a X en la línea 20 se efectúa un redondeo de la cifra llamado "redondeo de 4/5". Así, si la fracción tuviera un dígito igual a 5 (es decir mayor que 4; de ahí el nombre) en el lugar de las diezmilésimas, al agregar .0005 se incrementaría el dígito de la milésima en una unidad. El error cometido en este redondeo, después de efectuada la operación INT es como máximo de .0005 mientras que sin la línea 20 se podría cometer un error máximo de .001.

Ensayar otros valores para X (hay que asegurarse de que X*1000 no sea demasiado grande para la función INT).

Es fácil cambiar el programa para lograr redondear en un punto diferente. Por ejemplo para redondear X en el lugar de los cientos (2 dígitos a la derecha del punto decimal), se pueden cambiar las líneas 20 y 30 de este modo :

```
20 X = X + .005
30 X = INT (X * 100)/100
```

Un programa aplicación de la función INT puede ser la separación de los dígitos de un número decimal. Para realizarlo, tecleemos NEW y escribamos de nuevo el programa que convertía a un número X en una parte entera y otra decimal :

```
30 X = 3.14159
40 Y = INT(X)
50 Z = X - Y
60 PRINT " X= "; X
70 PRINT "Y = "; Y
80 PRINT " Z= "; Z
```

Si multiplicamos el valor Z por 10 entonces Z se convertirá en un número entero, además de tener una parte decimal : 1.41589. Podremos entonces tomar el valor entero y quitarle la parte decimal, dejando al dígito a mano izquierda sólo. Démosle el nombre de L al dígito a mano izquierda y veamos qué es lo que sucede. Añadamos :

```
 90 M = Z  10
100 L = INT(M)
110 PRINT "L ="; L
```

... y RUN.

La pantalla nos muestra :

```
X = 3.14159
Y = 3
Z = .141589
L = 1
```

Ahora podemos usar una iteración FOR-NEXT con el fin de quitarle unos cuantos dígitos más. Escribamos :

```
 95 FOR A = 1 TO 6
120 M = M - L
130 M = M * 10
```

```
140 NEXT A
```

... y RUN.

La impresión es :

```
X = 3.14159
Y = 3
Z = .141589
L = 1
L = 4
L = 1
L = 5
L = 8
L = 9
```

La pantalla muestra en forma descendente, todos y cada uno de los dígitos. Vamos a analizar las adiciones al programa (después de hacer un LIST).

En la línea 95 empieza una iteración FOR-NEXT de 6 pasos, uno para cada uno de los seis dígitos a la derecha del decimal.

La línea 120 origina un nuevo valor decimal a M (se trata tan sólo de una localización temporal de almacenamiento) al cortar la parte entera (poniendo los valores, M = 1.41589 - 1 = .41589).

La línea 130 hace lo mismo que hizo la línea 90 multiplica el nuevo valor decimal por diez a fin de convertir el dígito a mano izquierda en un entero y hacerlo capaz de ser utilizado por la función INT(M = .41589 *10 = 4.1589).

La línea 140 mueve el control de nuevo a la línea 95 para que se realice otra pasada a través del programa cortador.

LINEAS DE IMPRESION TEMPORALES

Hasta ahora se ha visto suficiente materia como para entender, que resulta tanto más difícil seguir el programa, cuanto más largo es. Sin embargo, existe una manera sencilla que puede resultar de gran utilidad.

Podemos poner "líneas de impresión temporales" en cualquier lugar de cualquier programa con el fin de seguirlo paso a paso en su ejecución. La computadora puede fácilmente saturarnos de datos, pero indicándole cuidadosamente qué es lo que queremos saber, podemos examinar los detalles internos de los cálculos. Agregar la siguiente línea :

```
92 PRINT " ## 92 M = "; M
```

... y RUN.

Los puntos esenciales de esta línea de "prueba", ó "bandera", son :

1. Está imprimiendo algo.
2. La impresión nos dice el número de la línea, lo que nos sirve de análisis y fácil localización para un borrado posterior.
3. Nos dice el nombre de la variable que se está observando en ese punto específico del programa.
4. Nos dá el valor de la variable en ese punto.

Esta "bandera" resulta de gran utilidad cuando se coloca en iteraciones FOR-NEXT. Escribamos :

```
97 PRINT " ## 97 A = "; A
```

... y RUN.

De esta manera la información temporal sale en la pantalla y se puede seguir el desarrollo en la ejecución del programa.

Para evitar que los datos se mezclen con la información, se pueden utilizar espaciamientos. Escribir :

```
92 PRINT "  ", " # 92 M = " ; M
97 PRINT "  ", " # 97 A = ";A
```

... y RUN.

Ahora, la lectura de datos y de información temporal resulta mucho más fácil. Vamos a colocar otros puntos en el programa. Escribamos :

```
125 PRINT "  "," "," "  #125 M = ";M
135 PRINT "  "," "," "  #135 M = ";M
```

... y RUN.

Ayudándose de la flecha ascendente, se puede estudiar la ejecución del programa paso a paso, obteniendo información complementaria en puntos del programa que ayudan a la comprensión del mismo.
Después de verificar, que el programa trabaja correctamente, se pueden borrar las banderas de información temporal puesto que ya han cumplido con el cometido a que estaban destinadas.

EJERCICIO 14-1 : Escribir este pequeño programa para encontrar el área de un círculo. (Primero teclear NEW).

```
1Ø  P = 3.14159
2Ø  PRINT "RADIO", "AREA"
3Ø  PRINT
4Ø FOR R = 1 TO 1Ø
5Ø  A = P* R * R
6Ø  PRINT R,A
7Ø  NEXT R
```

El área es igual a π multiplicado por el cuadrado del radio (esto es, el radio multiplicado por sí mismo). Ejecutarlo y asegurarse de que funciona.

Se pide : modificar el programa para obtener el resultado sin decimales, (añadiendo una línea 55).

EJERCICIO 14-2 : Cambiar la línea 55 del ejercicio anterior para que cada valor del área se redondee (hacia abajo), conservando la primera cifra decimal. Ejemplo :

RADIO	AREA
1	3.1

etc...

EJERCICIO 14-3 : Modificar la línea 55 del programa para que redondee (hacia abajo) el valor del área, de manera que tenga una precisión de dos decimales.

EJERCICIO 14-4 : Revisar la línea 55 para conseguir una precisión de 3 decimales en el cálculo del AREA.

Material Aprendido en el Capítulo 14

FUNCIONES	VARIOS
INT(X)	Banderas (o indicadores)

SOLUCION A LOS EJERCICIOS PROPUESTOS EN EL CAPITULO 14

RESPUESTA TIPO PARA EL EJERCICIO 14-1 :

El agregar la única línea siguiente da como resultado una impresión bonita y limpia, con todos los valores "redondeados" a su valor entero :

```
55  A = INT(A)
```

RESPUESTA TIPO PARA EL EJERCICIO 14-2 :

```
55  A = INT(1Ø * A)/1Ø
```

Cuando se multiplica 3.14159 por 1Ø, se convierte en 31.4159 El valor entero de 31.4159 es 31 y dividido por 1Ø es 3.1.

RESPUESTA TIPO PARA EL EJERCICIO 14-3 :

Este es casi demasiado sencillo.

```
55  A = INT(1ØØ * A)/1ØØ
```

RESPUESTA TIPO PARA EL EJERCICIO 14-4 :

Probablemente se haya tecleado :

```
55  A = INT(1ØØØ  * A)/1ØØØ
```

pero se ha obtenido un aviso de error
HOW?

Teóricamente el problema está bien, pero la computadora no trabaja. La explicación es simple : El error se debe a que la función INT(A) no admite valores de A mayores que 32737 y en nuestro problema hemos rebasado ampliamente esta cifra.

Con un BASIC NIVEL II, el problema habría funcionado sin complicaciones, pero el tener un NIVEL I nos obliga a pensar en otro tipo de solución para el problema.

```
55  A = INT(A) + INT((A - INT(A)) * 1ØØØ)/1ØØØ
```

CAPITULO 15

LA DECLARACION ON-GOTO

Escribir el siguiente programa :

```
 1Ø  INPUT "TECLEAR UN NUMERO ENTRE 1 Y 5" ;N
 2Ø  IF N = 1 GOTO 11Ø
 3Ø  IF N = 2 GOTO 13Ø
 4Ø  IF N = 3 GOTO 15Ø
 5Ø  IF N = 4 GOTO 17Ø
 6Ø  IF N = 5 GOTO 19Ø
 7Ø  PRINT "EL NUMERO QUE HA TECLEADO NO ESTA ENTRE 1 Y 5"
 99  END
11Ø  PRINT " N = 1"
12Ø  END
13Ø  PRINT " N = 2"
14Ø  END
15Ø  PRINT " N = 3"
16Ø  END
17Ø  PRINT " N = 4"
18Ø  END
19Ø  PRINT " N = 5"
```

NOTA :

Una línea de programa puede tener hasta 72 caracteres incluyendo el número de línea y los espacios. Como la pantalla tiene sólamente 64 caracteres, cuando se "lista" una línea de 72 caracteres, se ocupan dos líneas en la pantalla, pero sigue siendo una línea en el programa.

Ejecutar el programa asegurándose de que está libre de errores.

Aunque este programa funciona correctamente, existe una forma que lo simplifica. Esta forma es la ON-GOTO y para examinar cómo actúa, vamos a borrar las líneas 2Ø, 3Ø, 4Ø, 5Ø y 6Ø y escribir :

```
2Ø  ON N GOTO 11Ø, 13Ø, 15Ø, 17Ø, 19Ø
```

Si se ejecuta el programa, se obtienen los mismos resultados anteriores.

La declaración ON-GOTO es realmente muy sencilla. La línea 2Ø dice,

Si el valor entero de N(INTEGER) es 1 entonces vete (GOTO) a la línea 11Ø.
Si el valor entero de N(INTEGER) es 2 entonces vete (GOTO) a la línea 13Ø.
Si el valor entero de N(INTEGER) es 3 entonces vete (GOTO) a la línea 15Ø.
Si el valor entero de N(INTEGER) es 4 entonces vete (GOTO) a la línea 17Ø.
Si el valor entero de N(INTEGER) es 5 entonces vete (GOTO) a la línea 19Ø.
Si el valor entero de N(INTEGER) no es igual a ninguno de los números anteriores, entonces pasa a la siguiente línea.

La declaración ON-GOTO, tiene la declaración INT incorporada, que realmente actúa así :

```
2Ø  ON INT(N) GOTO .. . ETC.
```

Teclear los siguientes valores de N para demostrar este punto :

1.5

3.99999

Ø.999

5.999

6.ØØØ1

Existen muchos trucos que pueden ser empleados para sacar el máximo rendimiento a la declaración ON-GOTO. Por ejemplo, si se quisiera mandar a 15 diferentes lugares, obviamente no se puede escribir tantos números diferentes sobre una línea ON-GOTO, pero se pueden utilizar varias líneas, como sigue :

```
2Ø  ON N GOTO 11Ø, 13Ø, 15Ø, 17Ø, 19Ø
25  ON N-5 GOTO 21Ø, 23Ø, 25Ø, 27Ø, 29Ø
3Ø  ON N-1Ø GOTO 31Ø, 33Ø, 35Ø, 37Ø, 39Ø
```

... y se ponen las respuestas apropiadas en aquellos números de línea.

En la línea 25 se le ha restado 5 al número N; así cuando N valga 6, para la línea 25(N-5) valdrá 1 y mandará ir a la dirección 21Ø. En la línea 3Ø se ha usado el mismo procedimiento. Como acabamos de ver, N puede ser, bien una variable alfabética, o bien una expresión completa que, además, puede haber sido calculada previamente.

Debido a la incorporación de la función INT a la función ON-GOTO es muy común hacer la siguiente modificación :

```
5Ø  ON N +.2  GOTO 1ØØ, 2ØØ, 3ØØ  ETC...
```

De este modo se agrega una cantidad pequeña a N sin perder de vista que la declaración ON-GOTO contiene su propia función INT. Si N ha sido redondeada por ejemplo a 1.98 (en vez de un 2.ØØØ esperado), se le agregará Ø.2 haciendo a N 1.98 +.2 = 2.18 que gracias a la función INT incorporada se redondeará al valor deseado de 2.

Frecuentemente se le agregan a la N, para estos propósitos, valores entre .1 y .5 en programas escritos correctamente.

LA FUNCION SGN Y SUBRUTINAS

La función SGN examina cualquier número para ver si es negativo, cero ó positivo. Nos dice que el número es negativo proporcionando un (-1). Si el número es cero nos da un (Ø). Si es positivo, obtendremos un (+1). Es una función muy sencilla.

Desafortunadamente el BASIC NIVEL I no incluye la función SGN, pero mediante programa, es posible crear o simular funciones que no se tengan. Estas funciones elaboradas reciben el nombre de subrutinas.

Una subrutina es un programa corto pero muy especializado que se incorpora a un programa principal para satisfacer una necesidad especializada. El BASIC NIVEL 2 almacena muchas de ellas en memoria y pueden ser llamadas por una simple secuencia de letras. (En el NIVEL 1, existen algunas como INT).

En el apartado 6º de este manual, se encuentran una variada gama de subrutinas.

No hay suficiente memoria disponible en el NIVEL 1, como para guardar todas las subrutinas en la memoria, por lo que vamos a utilizar una subrutina de cinco líneas en vez de la función SGN para lograr lo mismo. Incluso disponiendo del BASIC NIVEL 2 se debe completar este capítulo para asegurarse de que se ha aprendido lo relacionado con las subrutinas.

Escribamos la subrutina que aparece en el apartado 6º :

```
30000  END
30800  REM * SGN(X)* INPUT X, OUTPUT T = -1,0 OR + 1
30810  IF X < 0 THEN T = - 1
30820  IF X = 0 THEN T = 0
30830  IF X > 0 THEN T = + 1
30840  RETURN
```

La línea 30000 constituye como se recordará, un bloque protector.

LLAMADA A UNA SUBRUTINA

La llamada a una subrutina se realiza con la declaración GOSUB # # # #. Esta declaración le dice a la computadora que vaya al número de línea indicado (# # # #) y una vez ejecutada la subrutina devuelve el control al punto del programa o línea siguiente a la declaración GOSUB de partida. El retorno se realiza por medio de una declaración (RETURN), que se coloca al final de la subrutina.

Para que se entienda mejor, vamos a combinar la función SGN (usando la subrutina del apartado 6) y la declaración GOSUB en el siguiente programa :

```
10  INPUT "TECLEAR CUALQUIER NUMERO"; X
20  GOSUB 30800
30  ON T + 2 GOTO 50, 60, 70
45  END
50  PRINT "EL NUMERO ES NEGATIVO"
55  END
60  PRINT "EL NUMERO ES CERO"
65  END
70  PRINT "EL NUMERO ES POSITIVO"
```

... y RUN.

Dar a X valores positivos, negativos y cero para comprobar que funciona.

Vamos a analizar el programa :

La línea 1Ø da entrada a cualquier número.
La línea 2Ø manda a la computadora a la línea 3Ø8ØØ mediante una declaración GOSUB.
Esto es diferente de un GOTO normal, ya que GOSUB devuelve el control, a la línea siguiente de aquella que manda ir a la subrutina.
La GOSUB no se terminará y no se alcanzará la siguiente línea de programa hasta encontrar RETURN.
Las líneas 3Ø8ØØ a 3Ø84Ø contienen la subrutina.
La línea 3Ø84Ø contiene el RETURN el cual devuelve el control a la línea 3Ø.
La línea 3Ø es una declaración ON-GOTO ordinaria, pero se le agrega 2 al valor de su variable, en este caso, T.
La línea 3Ø en realidad dice "Si T es 1 entonces vete (GOTO) a la línea 5Ø. Si es Ø entonces vete (GOTO) a la línea 6Ø, y si es +1 vete (GOTO) a la línea 7Ø". Agregándole un 2 a cada uno de estos valores los hemos convertido en 1, 2 y 3 que quedan así incorporados a la ON-GOTO.

Las líneas 45, 55 y 65 son bloques protectores de rutina.

En este interesante capítulo hay que destacar la importancia de los conceptos rutina-subrutina. Para completarlo, diremos que cualquier subrutina puede ser incorporada a un programa como rutina, en lugar de colocarse aparte de él. En este caso las líneas GOSUB y RETURN sobran y la subrutina sólo puede ser usada una vez. En esto último radica la importancia de las subrutinas, en que pueden ser utilizadas dentro de un programa un número ilimitado de veces; basta para ello con hacer llamadas GOSUB.

El valor de usar rutinas especiales como subrutinas es que algunas de ellas son excesivamente complejas para ser escritas sin error, y si cada una de ellas se escribe sólo una vez y es almacenada en cinta cassette, puede resultar más rápido, y se puede cargar en la computadora como primer paso en la creación de un nuevo programa.

EJERCICIO 15-1 : Eliminar las líneas 3Ø8ØØ a 3Ø84Ø del programa residente. Construir la rutina SGN dentro del programa de manera que funcione igual que llamando a la subrutina.

Material Aprendido en el Capítulo 15

FUNCIONES	DECLARACIONES	VARIOS
SGN(X)	ON-GOTO	Eliminación de errores
	GOSUB	Llamada a una subrutina
	ON-GOSUB	Rutinas
	RETURN	

SOLUCION A LOS EJERCICIOS PROPUESTOS EN EL CAPITULO 15

RESPUESTA TIPO PARA EL EJERCICIO 15-1 :

```
1Ø  INPUT "TECLEAR CUALQUIER NUMERO"; X
2Ø  REM * RUTINA SGN *
22  IF X < Ø THEN T = -1
24  IF X = Ø THEN T = Ø
26  IF X > Ø THEN T = +1
3Ø  ON T + 2 GOTO 5Ø, 6Ø, 7Ø
45  END
5Ø  PRINT "EL NUMERO ES NEGATIVO"
55  END
6Ø  PRINT "EL NUMERO ES CERO"
65  END
7Ø  PRINT "EL NUMERO ES POSITIVO"
```

CAPITULO 16

LECTURA DE DATOS

Hasta aquí hemos aprendido cómo introducir valores numéricos en el programa por dos métodos diferentes.
El primero es, dar el valor dentro del programa :

```
10  A = 5
```

El segundo es hacer uso de la declaración INPUT para introducir un número a través del teclado :

```
10  INPUT A
```

Una tercera forma consiste en utilizar la declaración DATA.
Escribamos el programa siguiente :

```
10  DATA 1,2,3,4,5
20  READ A,B,C,D,E
30  PRINT A;B;C;D;E
```

... y RUN.

La declaración DATA es en muchos aspectos similar al primer método en que la línea DATA es parte del programa. Es diferente, sin embargo, ya que cada línea DATA puede contener muchos números, cada uno de ellos separados por una coma. Cada parte de DATA debe ser leída por una declaración READ. Cada declaración READ puede leer un número determinado de partes del DATA si cada variable alfabética se separa por una coma.

La pantalla muestra que las 5 partes del dato en la línea 10 (los números 1,2,3,4,5), han sido leídas (READ) por la línea 20 (la cual asigna letras de la A a la E), e impresas por la línea 30.

Debemos tener presentes estas diferencias : Las líneas DATA pueden ser leídas únicamente por declaraciones READ.
Si escribimos varios datos a continuación de la declaración DATA, debemos separarlos mediante comas.
Los datos suministrados por el teclado pueden ser introducidos únicamente a través de declaraciones INPUT.

Las líneas DATA, siempre se leen de izquierda a derecha por las declaraciones READ; si existen varias líneas DATA, la lectura comienza por la de número de línea más bajo.

Las líneas DATA pueden colocarse en cualquier lugar del programa. Para comprobarlo :

1º) Pasemos la línea DATA de la 1Ø a la 25 y ejecutemos. El resultado no debe cambiar.

2º) Pasemos la línea DATA de la 25 a la 1ØØØ. El resultado tampoco cambiará.

Este hecho permite que distintos programadores utilicen estilos diferentes. Algunos colocan todas las líneas DATA al comienzo del programa, con el fin de que resulte rápido encontrarlas para cambiar datos. Otros las colocan todas al final fuera de donde podrían estorbar y donde existen más líneas disponibles para seguir agregando datos.
Algunos sitúan las líneas DATA a lo largo del programa, inmediatamente después de las líneas READ. El estilo que se use, no tiene consecuencias importantes, pero es conveniente adoptar uno determinado.

Vamos a ver ahora como trabaja la línea READ en medio de una iteración. Borremos el programa anterior mediante la instrucción NEW y tecleemos :

```
10  DATA 1,2,3,4,5
20   FOR N = 1 TO 5
30    READ A
40    PRINT A;
50   NEXT N
```

... y RUN.

El cambio de la posición de la línea 10 a la 25, no modifica en nada la ejecución.
Observemos que en cada iteración se lee y se escribe la letra A, pero su valor es diferente cada vez.
Esto es lógico porque los datos se van introduciendo sucesivamente y de uno en uno.

Cuando se agotan los datos de una línea DATA se comienza con la siguiente línea DATA, siempre que una declaración READ continúe solicitando la lectura de los mismos.

Cambiar la línea 20 del programa de la siguiente manera

```
20  FOR N = 1 TO 6
```

... y RUN.

En este caso, como la declaración READ pide la lectura de 6 datos y sólo tenemos 5 en la línea DATA, aparecerá un mensaje de error en la pantalla.

```
1 2 3 4 5 HOW?
30 READ A?
```

Cambiemos ahora la línea 20 de modo que el número de datos a leer sea menor que el de datos disponibles :

```
20  FOR N = 1 TO 4
```

... y RUN.

El programa se ejecuta normalmente con tal de que no se utilicen todos los datos disponibles y se pidan más. Debemos subrayar que cada dato de una declaración DATA, puede ser leído una sola vez por ejecución.

Como excepción de esto último, diremos que la declaración RESTORE permite releer los datos, debido a que a veces resulta necesario leerlos más de una vez en una sola ejecución. En el momento en que aparece en el programa un "RESTORE", todas las líneas DATA son restauradas a la situación de"no leídas", hayan sido leídas o no anteriormente, iniciándose de nuevo la lectura con el primer dato de la primera línea DATA. Cambiemos otra vez la línea 2Ø :

```
2Ø  FOR N = 1 TO 5
```

Introduzcamos :

```
35  RESTORE
```

... y RUN.

La pantalla imprime 5 veces 1, en lugar de 1,2,3,4,5 . ¿Por qué? . La línea 3Ø lee un 1 para A, pero la línea 35 restaura inmediatamente LA LINEA DATA A SU CONDICION ORIGINAL DE NO-LECTURA. Cuando la iteración FOR-NEXT obliga a pasar de nuevo por la línea READ en el ciclo siguiente, se lee otra vez la misma porción de información, que consiste en el mismo 1. Lo mismo sucede con todas las pasadas posteriores.

Las declaraciones READ y DATA son muy comunes. Se usa con menos frecuencia la declaración RESTORE (RESTAURAR).

VARIABLES ENCADENADAS O ALFANUMERICAS (ARREGLOS)

Hemos estado utilizando las letras de la A a la Z para indicar números. Se denominan VARIABLES NUMERICAS. En el BASIC NIVEL I se han reservado dos símbolos adicionales para representar VARIABLES ENCADENADAS. Estos son A$, y B$, que llamaremos "cadena A, y cadena B". Las variables encadenadas pueden ser asignadas para que indiquen Letras, Palabras y/o Combinaciones de letras, números y espacios de hasta 16 caracteres. Tecleemos NEW y a continuación escribamos :

```
10  INPUT "COMO TE LLAMAS"; A$
20  PRINT "HOLA,";A$
```

... y RUN.

Hasta ahora había 2 formas de imprimir palabras. La primera, aprendida en las primeras lecciones, era "depositando" palabras dentro de las declaraciones PRINT (se puede llamar "impresión de una cadena"). La segunda forma era introduciendo las palabras por medio de una declaración INPUT (se podía llamar "introducción de la cadena"). La tercera forma es la que acabamos de ver y que podemos llamar "lectura de una cadena".

Realicemos los cambios que se indican a continuación en el programa :

```
10  READ A$
20  DATA RADIO SHACK TRS-80
30  PRINT "VES MI ASTUCIA"; A$
```

... y RUN.

La pantalla muestra un error debido a que una variable encadenada sólo puede abarcar 16 caracteres.

El BASIC NIVEL I tiene dos variables encadenadas disponibles. Aprovechando estas dos variables podemos escribir de nuevo el programa para que imprima el nombre completo de la microcomputadora.

```
10  READ A$
15  READ B$
20  DATA RADIO SHACK,TRS-80
30  PRINT "VES MI ASTUCIA"; A$;"  ";B$
```

Analizando el programa :

La línea 20 contiene dos elementos DATA, separados por una coma.
La línea 10 lee el primero.
La línea 15 lee el segundo.
La línea 30 contiene cuatro expresiones de impresión. La primera imprime VES MI ASTUCIA, dejando un espacio detrás de la "A" ya que las variables encadenadas siempre consideran las letras juntas, permitiéndo la opción de insertar el espacio. La segunda imprime RADIO SHACK. La tercera impresión es el espacio comprendido entre las comillas. La última expresión es TRS-80.

EJERCICIO 16-1 : Diseñar un programa que produzca exactamente los mismos resultados, pero utilizando sólamente A$ y no B$. Sugerencia : Utilizar un bucle FOR-NEXT. La mayor parte del aprendizaje hasta aquí, está contenido en este sencillo programa. Debemos asegurarnos de haberlo entendido bien antes de continuar.

Recordando capítulos anteriores, en la modalidad de calculadora se veía la posibilidad de revisar el contenido de cada variable mediante la forma :

```
PRINT A;B;C;D;E; etc.
```

Podemos hacer lo mismo con nuestras dos nuevas cadenas. Escribamos :

```
PRINT A$;B$
```

¿Por qué aparece

TRS-80TRS-80?

El primer TRS-80 es sencillo, ya que, es la última cadena leída por A$. Lo mismo sucede con B$. Aunque B$ no fué utilizada para resolver el último problema, se usó en un ejemplo anterior y si la computadora no se ha apagado desde entonces, debe haberse mantenido su contenido en la memoria. Conviene no olvidar este pequeño detalle porque en programas complicados puede llevarnos a problemas de programación inexplicables.

En el nivel I no se pueden utilizar variables encadenadas, para tratamiento de textos porque sólo se dispone de 2. El nivel II permite hacerlo de forma directa. De todos modos, existe un procedimiento, aunque no tan sencillo, para conseguir el mismo propósito.

Escribamos :

```
10  S = 1
20  N = 0
30  INPUT  "TIENE SUFICIENTE EDAD PARA PODER VOTAR
    (S/N)";A
40  IF A = 1 THEN 60
50  PRINT "NO SE DESESPERE. EL TIEMPO PASA RAPIDO"
59  END
60  PRINT "MUY BIEN.¡NO OLVIDE REGISTRARSE!"
```

... y RUN.

Análisis :

La línea 1Ø fija el valor de la S igual a 1.
La línea 2Ø fija el valor de la N igual a Ø. Estos dos valores pueden inspeccionarse rápida y sencillamente tecleando PRINT S,N en la modalidad de calculadora.

La línea 3Ø da entrada a la respuesta de la pregunta, ya sea S o N (también aceptará SI y NO y cualquiera otras palabras que empiecen con S y N). "A" toma el valor de la S o de la N como se define en las líneas 1Ø y 2Ø.
La línea 4Ø prueba el valor de A, y si es 1, manda el control a la línea 6Ø. Si no es 1 (pero no necesariamente Ø), la línea 4Ø lleva a la línea 5Ø que imprime su mensaje.
La línea 59 es un END de protección con el fin de que la línea 6Ø no se imprima, si es que ya lo ha sido la línea 5Ø.

Le hemos dado cuidadosamente a S y a N los valores de 1 y Ø. Esto no significa que otras letras no puedan tener esos mismos valores. Eso no importa con tal de que solamente se oprima S ó N.
Como una protección parcial para impedir que el usuario oprima una letra incorrecta y accidentalmente llegue a una respuesta equivocada, podemos reforzar nuestro programa agregando estas líneas :

```
45  IF A = Ø THEN 5Ø
47  PRINT "POR FAVOR RESPONDA YA SEA CON UNA S O UNA N"
49  GOTO 3Ø
```

Análisis :

La línea 45 insiste en que para obtener la respuesta en la línea 5Ø, se debe teclear un cero. Sabemos que una N lo hará con toda seguridad. Otras letras podrán también tener el valor de cero.
La línea 47 nos da una respuesta en caso de "no pasar la prueba", advirtiéndolo al operador que tecleó algo distinto de SI o NO.

La línea 49 manda a la línea 3Ø con objeto de ofrecer una nueva oportunidad de oprimir la tecla correcta.

NOTA :

Ensayar un PRINT A;B;C;D; etc. para comprobar si cualquier otra variable alfabética está almacenando un valor 1 o cero.

EJERCICIO 16-2 : Diseñar y escribir un programa sencillo que pregunte al usuario por lo menos cinco preguntas y en el proceso entable una pequeña conversación con él. Lógicamente para este ejercicio no se da la solución puesto que las posibilidades son infinitas.

Material Aprendido en el Capítulo 16

DECLARACIONES	VARIOS
READ	Variables Encadenadas (Arreglos) A$,B$
DATA	Variables numéricas
RESTORE (RESTAURAR)	(S/N) (enseñándo a la TRS-80 a responder SI ó NO).

SOLUCION A LOS EJERCICIOS PROPUESTOS EN EL CAPITULO 16

RESPUESTA TIPO PARA EL EJERCICIO 16-1 :

```
 3 PRINT  "VES MI ASTUCIA";
 5 FOR N = 1 TO 2
1Ø READ A$
2Ø DATA RADIO SHACK, TRS-80
3Ø PRINT A$;"   ";
4Ø NEXT N
```

Análisis :

La línea 3 imprime la primera parte, dejando un espacio para la impresión del A$ siguiente y tiene un punto y coma final con el fin de que se suprima el retorno del carro.

La línea 5 establece una iteración FOR-NEXT de dos pasadas.

La línea 1Ø lee RADIO SHACK.

La línea 2Ø contiene las dos cadenas de datos, separadas por una coma.

La línea 3Ø imprime radio Shack y un espacio en la primera pasada de la iteración N. Observar que el punto y coma final se encuentra de nuevo para suprimir el retorno del carro.

La línea 4Ø devuelve el control a la línea 5 (la iteración); la segunda pasada a través de la iteración, imprime TRS-80 y otro espacio en blanco, (que no produce efecto alguno porque el programa ha finalizado).

CAPITULO 17

FUNCION "VALOR ABSOLUTO"

Después de algunos capítulos bastantes densos, vamos a ver ahora un capítulo muy breve, pero no por ello exento de interés.
Nos referimos a la función " valor absoluto".

El VALOR ABSOLUTO de cualquier número es ese mismo número pero sin signo, ya fuese éste positivo o negativo. Es decir :

Si X es menor que Ø, entonces ABS(X) = (-1) * X
Si X = Ø entonces ABS(X) = X = Ø
Si X es mayor que Ø, entonces ABS(X) = X

Escribamos :

```
1Ø  INPUT  "TECLEAR CUALQUIER NUMERO POSITIVO O NEGATIVO";X
2Ø  Y = ABS(X)
3Ø  PRINT "X", "Y"
4Ø  PRINT X,Y
```

... y RUN, dando diferentes valores numéricos, positivos y negativos.

Sin importar que número se haya introducido (como X), su valor absoluto "Y" es el mismo número pero sin signo.

Material Aprendido en el Capítulo 17

FUNCIONES

ABS(X) = valor absoluto de X

CAPITULO 18

BASIC TAQUIGRAFICO DEL TRS-80

El BASIC del microordenador TRS-80 posee la característica de que se pueden editar los programas en formas abreviadas. Si no se han usado estas formas hasta ahora, se debe a que se pretendía ofrecer un conocimiento del BASIC usual, para posibilitar el intercambio de información con los distintos usuarios de este lenguaje.

Casi cualquier INSTRUCCION, DECLARACION y FUNCION tiene una notación en taquigrafía mucho más breve y sencilla, efectuando, sin embargo, el mismo cometido. La lista completa se encuentra al final del último capítulo. He aquí una lista de los estudiados hasta el momento.

NEW = N.
LIST = L.
RUN = R.
PRINT = P.
MEM = M.
STOP = ST.
CONT = C.
THEN = T.
END = E.
ABS = A.
RESTORE = REST.
GOTO = G.

INPUT = IN.
FOR = F.
NEXT = N.
CSAVE = CS.
CLOAD = CL.
STEP = S. (cuando se utilice con FOR-NEXT)
TAB = T. (cuando se utilice con PRINT)
INT = I.
DATA = D.
READ = REA.

NOTA :

Ahora podemos interpretar la prueba realizada en el Capítulo 1 después de encender la microcomputadora.

Tecleamos entonces P.M. solicitando de este modo la impresión del número de posiciones libres de memoria.

Además, el interpretador de Radio Shack permite poner más de una declaración en cada línea numerada, separándolas por un (:). Por ejemplo, una iteración cronometrada como :

```
1ØØ  FOR N = 1 TO 5ØØ
11Ø  NEXT N
```

se convierte en ...

```
1ØØ  FOR N = 1 TO 5ØØ:NEXT N
```

... más corto aún...

```
1ØØ F.N=1TO5ØØ:N.N
```

... aún menos espacio ...

```
1ØØF.N=1TO5ØØ:N.N
```

La última línea puede ser introducida a la computadora tal como se muestra sin dejar espacio libre entre el número de línea y la primera letra, aunque la computadora automáticamente introducirá un espacio cuando se le ordene un LIST. Operando como acabamos de indicar, se conserva el espacio extra que hemos venido introduciendo hasta la fecha.

Como puede apreciarse, la combinación entre la taquigrafía Radio Shack y las líneas de declaraciones múltiples, pueden llevarnos a una reducción importante de espacio en los programas. Como contrapartida, el juntar tanto la información dificulta su lectura y comprensión; por lo tanto en la mayor parte de este manual evitaremos las líneas de declaraciones múltiples. Sin embargo, cuando realmente se quiera ahorrar espacio, ésta será la única forma de hacerlo.

Al llegar a este punto hemos de hacer una observación importante : Debemos tener un cuidado especial cuando utilicemos la declaración IF-THEN en líneas de declaraciones múltiples. Para ilustrar este hecho, veamos qué ocurre al ejecutar el siguiente programa para diferentes valores de X.

```
10  IN.  "TECLEAR UN NUMERO"; X
20  IF X = 3 THEN 50:G.70
30  P. "COMO LLEGO HASTA AQUI?"
40  END
50  P. "X =3"
60  END
70  P."NO PUEDE LLEGAR DESDE ALLA HASTA AQUI".
```

... y RUN.

La línea 20 es ilegal. Si X vale 3 el control se transfiere a la línea 50, pero si dicha condición no se cumple, el control pasa a la línea 30. Así pues, nunca se tiene en cuenta la segunda declaración de la línea 20.

En general, al utilizar líneas de declaraciones múltiples, se debe tener en cuenta :

a) Cuando se quiera utilizar una declaración IF-THEN (ó un ON-GOTO), para indicar una prueba que se efectúa dentro de una línea declarativa múltiple, deberá ser la última declaración en esa línea.

b) No se puede mandar el control a ningún punto de la línea declarativa múltiple con excepción de su primera declaración. Observar la línea 20 del programa residente. Aún cuando el G.70 fuese legal en esa línea, no existe forma alguna de direccionarlo, ya que, comparte el mismo número de línea que la primera declaración de la línea.

Sólo la primera declaración podrá ser direccionable mediante un GOTO ó un IF-THEN.

c) Las líneas DATA, no pueden existir en líneas que contengan otras declaraciones.

EJERCICIO 18-1 : Escribir de nuevo cualquiera de los programas de capítulos anteriores, usando líneas de declaraciones múltiples así como, las expresiones taquigráficas de Radio Shack. Utilizar el P.M. para averiguar el número de posiciones de memoria disponibles. Resolver de nuevo el problema de manera que se use la menor memoria posible.

Material Aprendido en el Capítulo 18

FUNCIONES	VARIOS
IF (condición) (declaración) con THEN omitido.	Dialecto taquigráfico NIVEL I
	Líneas Declarativas múltiples.

CAPITULO 19

LA FUNCION RND (NUMERO ALEATORIO)

Un número al azar, es un número cuyo valor no se puede predecir. Un generador de números al azar es aquel que extrae uno determinado entre un número finito de posibilidades. El BASIC NIVEL I posee un generador de este tipo mediante la forma :

```
N = RND(X)
```

Donde N es un número al azar.
RND es una abreviatura y simboliza al azar (random).
X es un número de control que podrá ser escrito entre los paréntesis o traído como variable de algún otro lugar del programa.

Escribamos :

```
5Ø  PRINT RND(Ø)
```

... y RUN. Si se ejecuta por lo menos 10 veces, se observa que :

1. Cada vez ha aparecido un número diferente.
2. Todos los números están entre Ø y 1.
3. Los números pequeños han sido expresados en notación exponencial.

Una manera más cómoda de extraer los 10 números es mediante el bucle :

```
4Ø  FOR N = 1 TO 1Ø
5Ø  PRINT RND(Ø)
6Ø  NEXT N
```

Ahora vamos a poner un punto y coma al final de la línea PRINT e incrementar el bucle a 90 pasadas :

```
40  FOR N = 1 TO 90
50  PRINT RND(0);
60  NEXT N
```

... y RUN.

De esta manera es bastante apreciable el carácter aleatorio de los números obtenidos, y podemos intuir que la función RND va a ser muy útil para la mayoría de los programas de juegos.

VARIANTES DE RND

La RND(Ø) con la que se ha trabajado, es el generador tradicional de números al azar. Con ayuda de la función INT y algo de matemáticas se puede construir una RND de números aleatorios enteros. No vamos a estudiar esta técnica porque el BASIC TRS-80 incluye esta forma de operar en su propia RND. Comprobémoslo escribiendo :

```
50  PRINT RND(15)
```

... y RUN. Resumiendo :

1. Si el número dentro del paréntesis (o su valor entero) es cero, los números generados estarán entre Ø y uno.
2. Si el número entre paréntesis es mayor o igual a uno, los números generados serán de 1 al valor entero de aquel número (inclusive).
3. En el BASIC NIVEL I, el valor más alto permisible será 32767.

Para demostrar una vez más el carácter aleatorio de la función RND, vamos a estudiar un programa basado en el conocido juego de tirar la moneda al aire y comprobar cuántas veces sale cara y cuántas cruz. Cuantas más veces se tira la moneda, tanto más nos acercamos al 50% de caras o cruces.

Escribir este programa que simula la tirada de una moneda en taquigrafía Radio Shack y utiliza líneas declarativas múltiples sólo por practicar un poco. Teclearlo con mucho cuidado para evitar errores :

```
 1Ø  H = Ø:T=Ø:P
 2Ø  IN. "CUANTAS VECES TIRAREMOS LA MONEDA";F:CLS
 3Ø  P. "AGUARDE MIENTRAS TIRO LA MONEDA....."
 4Ø  F.N = 1 TOF:X=RND(2):ONXG.6Ø,7Ø
 5Ø  P. "FALLO! NO HA SIDO NI 1 NI 2":END
 6Ø  H=H + 1:G.8Ø
 7Ø  T=T+1
 8Ø  N.N:P.:P.:P.:P
 9Ø  P.  "CARAS", "CRUCES", "TOTAL DE TIRADAS":P.:P.H,T,F
1ØØ  P. 1ØØ  H/F; "%", 1ØØ  T/F; "%":P.:P.:P
```

... y RUN.

Tirar la moneda 10, 50, 100, 1000...etc. veces, hasta convencerse totalmente de que el generador de números al azar, produce verdaderamente números aleatorios.

Análisis del programa :

La línea 1Ø está constituida por tres declaraciones. La primera inicializa a H(CARAS) en cero.

La segunda fila T(para las CRUCES) igual a cero. La tercera inserta un espacio en la impresión de salida con un PRINT(P).

La línea 2Ø tiene dos declaraciones. 1- Da entrada al número deseado de tiradas. 2- Limpia la pantalla (CLS) para empezar la siguiente línea de impresión en la parte superior de la misma.

La línea 3Ø imprime una declaración o mensaje de "Espera".

La línea 4Ø consta de tres declaraciones. 1- Inicia una iteración FOR-NEXT que corre "F" veces. 2- Es el generador RND(X). Le hemos dicho que genere enteros entre 1 y 2, y por supuesto ésto lo restringe tan sólo a los números 1 y 2. La cara será el "1" y la cruz será el "2". 3- Una prueba ON-GOTO manda con X = 1 a la línea 6Ø que es donde se cuentan "las caras", y con X = 2 a la línea 7Ø que es donde se cuentan "las cruces". Observar que esta prueba es la última declaración de la línea declarativa múltiple.

La línea 5Ø es una línea en caso de error. Si X es diferente de uno ó dos (1 ó 2), se imprimirá el mensaje de error y la ejecución finalizará.

La línea 6Ø utiliza H como contador. La línea 1Ø inicializa H en cero, y cada vez que la prueba ON-GOTO manda el control a esa línea debido a que X =1, se incrementa (en 1) H y guarda la cuenta del número de "caras". La segunda declaración manda el control a la línea 8Ø que establece un nuevo ciclo del bucle. Cuando se han dado F vueltas, la línea 8Ø se dirige hacia las cuatro declaraciones PRINT(P.) vacías, devolviendo además el control a la línea 4Ø.

Si la línea 4Ø genera otro número al azar (1 ó 2) y este número es X = 2, entonces la ON-GOTO manda el control a la línea 7Ø.

La línea 7Ø cuenta las cruces, pasando luego a NEXT N en la 8Ø.

En la línea 9Ø, se imprimen los títulos, luego la línea en blanco y luego los valores de HT y F.

La línea 1ØØ calcula e imprime el porcentaje de caras y cruces, y por último imprime 3 líneas en blanco para que la presentación sea más adecuada.

GENERADORES SIMULTANEOS

Es posible generar más de un número al azar usando más de un generador en el programa. Esto tiene valor especial cuando los rangos de los generadores son diferentes, pero es de utilidad incluso cuando los rangos son los mismos.

Para el estudio de este apartado vamos a simular una tirada de 2 dados mediante programa. Cada dado tiene 6 caras, poseyendo cada cara una puntuación distinta comprendida entre 1 y 6. Cuando se tiran los dados, se suman el número de puntos obtenidos. Los números inferior y superior deben ser el 2 y el 12 respectivamente. Vamos a simular un generador de números al azar distinto para cada dado dándole a cada uno un rango de 1 a 6 y llamando a los dados A y B.

Escribamos NEW y luego :

```
5Ø  A=RND(6):B=RND(6):N=A+B
6Ø  P.N
```

... y RUN. Ejecutémoslo varias veces .

Como puede apreciarse, todos los números que imprime estan entre cero y 12. La suma se corresponde fielmente con la suma de la tirada de dos dados convencionales; decimos esto porque se podía haber generado un número aleatorio entre 2 y 12 que no cumpliera con el requisito, por ejemplo :

```
5Ø P.RND(11)+1
```

La explicación del por qué este generador no simula una tirada de dados es que las posibilidades de que salga un 2, un 5 ó un 7 con los dados son distintas, mientras que con este tipo de generador, son iguales.

En su forma más simple, el juego consiste en lo siguiente :

1. El jugador tira los dos dados. Si obtiene una suma de 2, de 3 ó de 12 en la primera tirada, pierde y el juego ha terminado.
2. Si el jugador saca 7 u 11 en la primera tirada (llamado natural) gana y el juego ha terminado.
3. Si saca cualquier otro número, éste se convierte en la puntuación del jugador. Deberá seguir tirando hasta que, o bien consiga la misma puntuación y gane, o bien, saque un 7 y pierda.

En el apartado 3º se muestra un programa de juego de dados. Referirse a él para estudiar detenidamente sus pasos. Una forma de aprender a programar es el estudio de programas terminados.

Los números al azar son impredecibles, los obtenidos por las computadoras que funcionan adecuadamente no lo son. Así es que, ¿cómo podemos obtener números al azar de una computadora? No los obtenemos; obtenemos números seudo-aleatorios. Cada vez que se utiliza la función RND, la computadora hace uso internamente de "un número semilla" para producir el número al azar deseado.

Material Aprendido en el Capítulo 19

FUNCIONES	VARIOS
RND(Ø) para números al azar mayores que Ø y menores que 1.	Azar vs Seudo-azar
RND(N) para números al azar entre 1 y N.	

CAPITULO 20

DIBUJO SOBRE LA PANTALLA DE VIDEO

A través de este capítulo, se estudiarán los procedimientos y capacidades básicas de dibujo en la pantalla de video. Partiendo de un pequeño bloque luminoso, se pueden ir creando sistemas de dibujo, que permitirán poner a prueba la imaginación del programador.

Veamos ahora dos instrucciones gráficas :

SET (Sitúa) Enciende un determinado bloque luminoso en la pantalla.

RESET (Vuelve a la situación inicial). Apaga un determinado bloque luminoso en la pantalla.

Para posibilitar el dibujo sobre la pantalla de video, se divide ésta en un número de secciones, (observar la hoja de trabajo del display de VIDEO que se encuentra en la página siguiente). Cada sección es un bloque rectangular de dos unidades de ancho por cuatro de alto y cada uno posee su propia dirección.

Por ejemplo la parte :

```
SET(55,32)
```

Significa "ilumina" el bloque situado en la intersección de la fila "X" 55 y la columna "Y" 32.

La X es la dirección horizontal que se cuenta a partir de la parte izquierda de la pantalla. La Y es la dirección vertical, y se cuenta en forma descendente a partir de la parte superior de la pantalla. Por lo tanto comienza en la esquina superior a mano izquierda.

Escribamos :

```
5Ø SET(55,32)
```

Borremos la pantalla y ejecutemos.

La luz se enciende exactamente en ese punto. Para cerciorarse, revisar la hoja de trabajo del display de video para encontrar la dirección del bloque luminoso.

Escribamos cuidadosamente :

```
5Ø RESET(55,32)
```

... y RUN.

Como podemos ver el bloque luminoso ha desaparecido. El efecto es el mismo que si hubiésemos instalado un interruptor para encender y apagar.

Escribamos de nuevo :

```
5Ø SET(55,32)
```

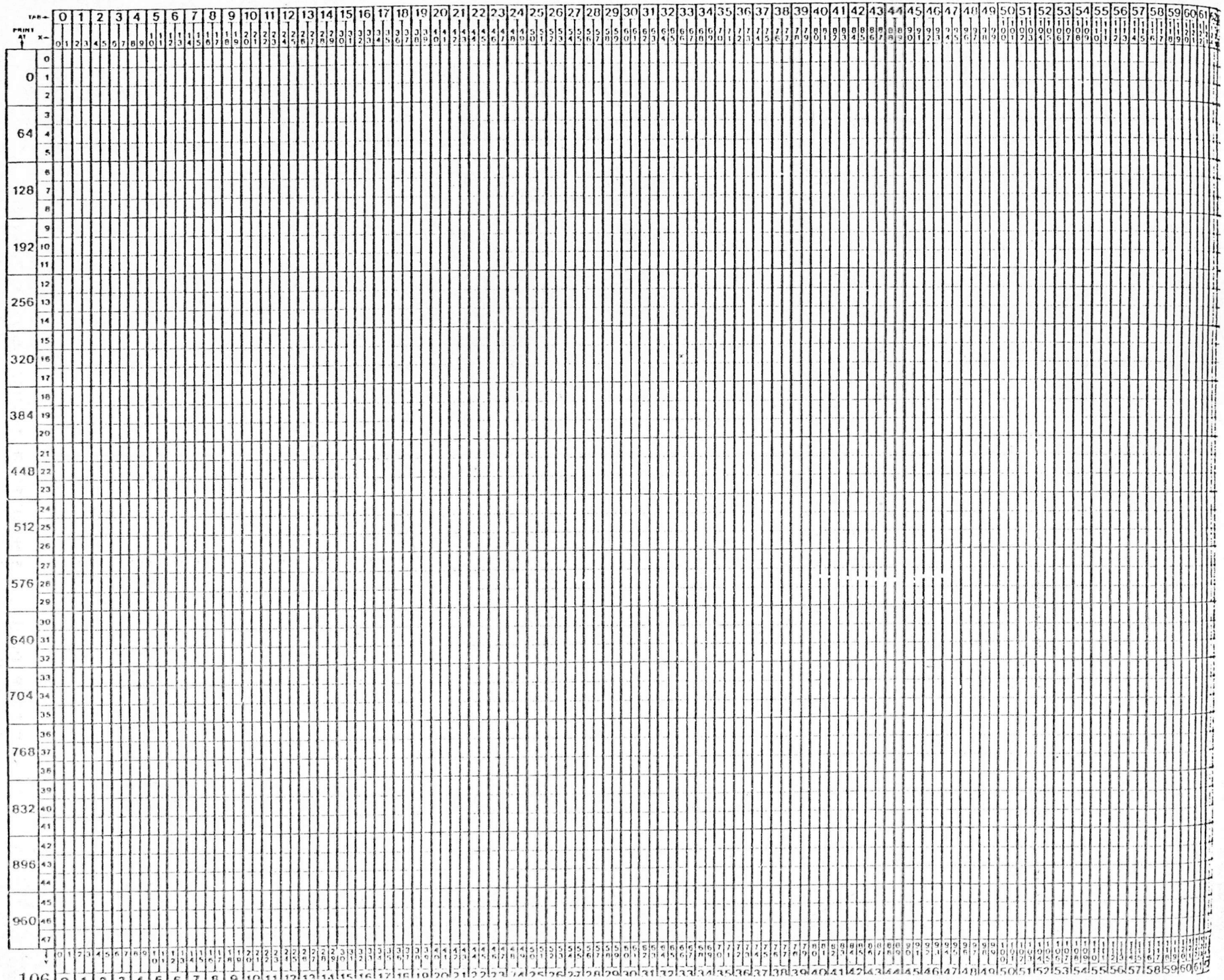
y ejecutémoslo 5 veces seguidas. Escribamos luego 5Ø RESET(55,32) y ejecutemos.

Como se aprecia, la luz no se ha apagado. La explicación está en que el retorno del carro después de la 5ª ejecución, ha movido la pantalla hacia arriba con lo cual ha dejado al bloque luminoso en una dirección distinta a la original, y sólamente si se encuentra en una dirección determinada se ilumina o se apaga. Las direcciones de la pantalla nunca se mueven.

Llegamos, por tanto, a la conclusión de que podemos controlar el encendido o apagado de cada bloque de la pantalla, controlando sus direcciones con declaraciones SET y RESET.

[redacted]oja[redacted]e [redacted]baj[redacted]pa[redacted]a [redacted]ta[redacted]de[redacted]de[redacted]TR[redacted]0 [redacted]

TITULO _____ PROGRAMADOR _____ COMENTARIOS _____ PAGINA _____ DE _____



INTERMITENCIAS

Anteriormente hemos visto que disponíamos de un interruptor de encendido y apagado. Ahora vamos a crear un bloque luminoso intermitente. Escribamos y ejecutemos :

```
1Ø CLS
2Ø X = 6Ø
3Ø Y = 25
4Ø SET(X,Y)
5Ø RESET(X,Y)
6Ø GOTO 4Ø
```

DIRECCIONES DE LOS BLOQUES DE LA PANTALLA DE VIDEO

Como puede apreciarse en la hoja de trabajo de la pantalla de video, existen 128 direcciones horizontales de bloques luminosos, numerados del cero al 127. El cero está en el extremo izquierdo, el 64 hacia la mitad y el 127 en el extremo derecho.

En la dirección vertical existen 48 direcciones de bloques luminosos numerados del Ø al 47. El cero se encuentra en la parte superior y el 47 en la parte inferior.

La declaración "SET(X,Y)" ilumina el bloque que se encuentre en la posición X-ésima, contada de izquierda a derecha en dirección horizontal y en la posición Y-ésima contada de arriba a abajo. El RESET trabaja de la misma forma.

Escribamos este programa que ilumina el bloque que se quiera escoger :

```
1Ø INPUT "LA DIRECCION HORIZONTAL (Ø a 127) ES";X
2Ø INPUT "LA DIRECCION VERTICAL (Ø a 47) ES";Y
3Ø CLS
4Ø SET(X,Y)
```

Ejecutémoslo para varios valores de X e Y.

¿Qué sucede si X = 15Ø ó Y = 6Ø? . Hagamos la prueba.

Cuando la dirección excede de los valores máximos bien sea de X ó de Y, comienza la cuenta en otra línea a partir de estos valores máximos.

PROGRAMAS GRAFICOS

Es probable haber observado que si el bloque se enciende en la esquina superior izquierda, el READY y el apuntador > lo destruyen. Probar con X = 6 e Y = 6; luego con X =4 e Y =4. Podemos evitar este problema evitando que regrese el control del apuntador agregando :

```
99 GOTO 99
```

Esta línea al final del programa, trabaja como un bucle sin fin. Para salir del bucle, hay que oprimir la tecla [BREAK.] Debe colocarse un bucle sin fin al final de todo programa gráfico. Puede efectuarse una prueba con valores tales como : X = Ø, Y = Ø; X = 4, Y = 4. Recordar que hay que oprimir la tecla [BREAK] para detener un programa gráfico.

Mientras que existe una tecla ([CLEAR]), que reajusta todos los bloques sobre la pantalla llevándolos a la situación de apagados, no tenemos ninguna otra similar para iluminarlos. Sin embargo, podemos escribir un programa que ilumine la pantalla entera. El programa utiliza un borrador (CLS), dos bucles FOR-NEXT y uno más sin fin.

Escribamos y ejecutemos :

```
1Ø CLS
2Ø FOR X = Ø TO 127
3Ø  FOR Y = Ø TO 47
4Ø   SET(X,Y)
5Ø  NEXT Y
6Ø NEXT X
99 GOTO  99
```

El programa cubre la pantalla de izquierda a derecha. Ahora vamos a rediseñarlo para que empiece en la parte superior y vaya cubriendo la pantalla hacia abajo.

Escribamos y ejecutemos :

```
1Ø CLS
2Ø FOR Y = Ø TO 47
3Ø  FOR X = Ø TO 127
4Ø   SET(X,Y)
5Ø  NEXT X
6Ø NEXT Y
99 GOTO 99
```

Ahora podemos hacer que se cubra de abajo a arriba :
Solución :

```
1Ø CLS
2Ø FOR Y = 47 TO Ø step -1
3Ø  FOR X = Ø TO 127
4Ø   SET (X,Y)
5Ø  NEXT X
6Ø NEXT Y
99 GOTO 99
```

Y ya sólo falta que el programa cubra la pantalla de izquierda a derecha.

Solución :

```
1Ø CLS
2Ø FOR X = 127 TO Ø STEP -1
3Ø  FOR Y = Ø TO 47
4Ø   SET(X,Y)
5Ø  NEXT Y
6Ø NEXT X
99 GOTO 99
```

EJERCICIO 20-1 : Escribir un programa que permita dibujar sólamente en una pequeña parte de la pantalla, introduciendo los números de los bloques inicial y final, correspondientes a sus direcciones horizontal y vertical .

Ahora podemos escribir sólamente líneas. Primero teclearemos NEW y luego escribiremos el programa correspondiente a una línea horizontal y lo ejecutaremos:

```
1Ø INPUT "DIRECCION VERTICAL (Ø a 47)"; Y
2Ø CLS
3Ø FOR X = Ø TO 127
4Ø  SET(X,Y)
5Ø NEXT X
99 GOTO 99
```

El programa correspondiente a una línea vertical corresponde al enunciado siguiente. Escribámoslo y ejecutémoslo :

```
1Ø INPUT "DIRECCION HORIZONTAL (Ø a 127)"; X
2Ø CLS
3Ø FOR Y = Ø TO 47
4Ø SET(X,Y)
5Ø NEXT Y
99 GOTO 99
```

Ahora nos proponemos modificar este último programa de manera que se pueda introducir, mediante un INPUT, la dirección inicial vertical y la longitud (en bloques) de la línea.

Solución :

```
12 IN. "LA DIRECCION VERTICAL INICIAL #(Ø A 47) ES";V
14 IN. "CUANTOS BLOQUES VERTICALES DESEA LLENAR";A
16 IF V + 1 < 48 GOTO 2Ø
18 PRINT "SON DEMASIADOS BLOQUES VERTICALES"
19 END
3Ø FOR Y = V TO V + A
```

Como ya se ha aprendido a dibujar líneas, ahora podemos dibujar cuadrados y rectángulos. El siguiente programa forma un rectángulo. Después de teclear NEW, escribamos :

```
 1Ø INPUT "PUNTO HORIZONTAL INICIAL (Ø A 127)";X
 2Ø INPUT "PUNTO VERTICAL INICIAL (Ø A 47)";Y
 3Ø INPUT "LONGITUD DE CADA LADO (EN BLOQUES)-- (Ø A 47)";K
 4Ø CLS
 5Ø FOR L = X TO X + K
 6Ø  SET(L,Y)
 7Ø  SET(L,Y + K)
 8Ø NEXT L
 9Ø FOR M = Y TO Y + K
1ØØ  SET (X,M)
11Ø   SET (X + K, M)
12Ø NEXT M
999 GOTO 999
```

... y RUN.

EJERCICIO 20-2 : Debido a que los bloques básicos son rectángulos de 2 por 4 y no cuadrados, con el programa anterior se obtienen rectángulos. Se propone cambiar el programa para obtener cuadrados. La solución se encuentra al final

del capítulo pero se recomienda intentarlo uno mismo.

OTROS TIPOS DE PROGRAMAS GRAFICOS

Todo nuestro trabajo relacionado con las gráficas hasta aquí ha consistido en dibujar líneas blancas sobre una pantalla oscurecida. Podemos hacer lo contrario, "pintando" inicialmente la pantalla de blanco, y posteriormente oscurecer las áreas deseadas con un RESET. En este programa, por ejemplo, se escribe una línea oscura sobre una pantalla con su fondo blanco. Escribamos :

```
10 IN. "POSICION VERTICAL (0 A 47)";Y
20 CLS
30 FOR X = 0 TO 127
40  FOR J = 0 TO 47
50  SET(X,J)
60  NEXT J
70 NEXT X
80 FOR X = 0 TO 127
90   RESET(X,Y)
100 NEXT X
999 G.999
```

... y RUN. Si se desea se pueden efectuar modificaciones sencillas, para dibujar en "positivo" y "negativo".

Podemos dibujar más líneas cambiando tan sólo las direcciones de X y de Y de SET en el bucle FOR-NEXT. Escribamos el programa siguiente para dibujar líneas diagonales.

```
10 INPUT "PUNTO HORIZONTAL INICIAL (0 A 127)";X
20 INPUT "PUNTO VERTICAL INICIAL (0 A 47)";Y
30 INPUT "LONGITUD DE LA DIAGONAL";K
40 CLS
```

```
5Ø FOR L = Ø TO K
6Ø  SET (X + L, Y + L)
7Ø NEXT L
99 GOTO 99
```

Una vez obtenida la línea diagonal, podemos formar un triángulo rectángulo agregando :

```
7Ø SET(X,Y +L)
8Ø SET(X + L, Y + K)
9Ø NEXT L
```

ó

```
7Ø SET(X + K, Y +L)
8Ø SET(X + L, Y)
9Ø NEXT L
```

La diferencia entre los dos fragmentos de programa añadidos, es que las imágenes son simétricas una de otra.

En todos los programas gráficos que se acaban de mostrar, se podía haber hecho que las líneas fueran punteadas introduciendo un STEP (incremento) diferente de 1 dentro de los bucles FOR-NEXT. Por ejemplo, podemos obtener una línea horizontal punteada con :

```
1Ø INPUT "DIRECCION VERTICAL (Ø a 47)";Y
2Ø INPUT "TAMAÑO DEL INCREMENTO";S
3Ø CLS
4Ø FOR X = Ø TO 127 STEP S
5Ø SET(X,Y)
6Ø NEXT X
99 GOTO 99
```

Ejecutemos este programa para varios valores de S. Observaremos que a medida que se incrementa S, se escribe la línea con mayor velocidad (ya que la computadora tiene menos puntos que iluminar). En realidad esta es la forma en que se produce una imagen de televisión, ya que también allí se escribe de unidad en unidad pero a tal velocidad que no es apreciado por el ojo humano.

Realicemos los siguientes cambios en el programa que nos ocupa :

```
55 RESET(X, Y -1)
7Ø Y = Y + 1
8Ø GOTO 4Ø
```

Si S es pequeña se puede ver la formación y borrado de las líneas. Pero si S es grande, (probar con S = 10), la línea parece moverse y casi no se aprecia el borrado.

En el siguiente programa se va a dibujar un punto sobre la pantalla, de manera que se mueva hacia abajo (también se podía haber hecho el movimiento hacia arriba).

```
1Ø INPUT "PUNTO HORIZONTAL INICIAL (Ø a 127)";X
2Ø INPUT "PUNTO VERTICAL INICIAL (Ø a 47)";Y
3Ø CLS
4Ø RESET(X,Y -1)
5Ø SET(X,Y)
6Ø Y = Y + 1
7Ø GOTO 4Ø
99 GOTO 99
```

La orden RESET borra simplemente el bloque luminoso del ultimo SET. Si se omite esta instrucción, el programa dibuja una línea.

Un detalle a tener en cuenta es que el SET ó RESET no trabaja con coordenadas negativas. Comprobémoslo escribiendo :

```
40 RESET(X, Y - 1)
```

Si se elige un valor de Y de entrada igual a cero, Y toma el valor -1, y el programa no se ejecuta. Veamos qué sucede si realizamos el siguiente cambio en la línea 40 :

```
40 RESET(X, Y + 47)
```

El programa se ejecuta ahora, eligiendo un punto de partida de Y = 0 .
También se puede mover el punto hacia la derecha, haciendo :

```
10 INPUT "PUNTO HORIZONTAL INICIAL (90 a 127)"; X
20 INPUT "PUNTO VERTICAL INICIAL (0  a 47)";Y
30 CLS
40 RESET (X+127,Y)
50 SET(X,Y)
60 X = X + 1
70 GOTO 40
99 GOTO 99
```

Observemos que se ha utilizado el mismo sistema anterior para evitar valores negativos de Y.

¿Qué sucede si se cambia la línea 40 por la siguiente?

```
40 RESET(X+126,Y)
```

Ejecutemos de nuevo. ¿Y si se cambia por esta otra?

```
40 RESET(X+125,Y)
```

Como puede apreciarse, el punto se hace mayor.

EJERCICIO 20-3 : Cambiar los últimos dos programas de manera que consigan desplazar el punto hacia arriba y hacia la izquierda respectivamente.

Ahora hagamos que el punto se desplace hacia abajo hasta que choque con una barrera. El programa es :

```
1Ø INPUT "PUNTO HORIZONTAL INICIAL (Ø a 127)";X
2Ø INPUT "PUNTO VERTICAL INICIAL (Ø a 47)";Y
3Ø INPUT "BARRERA INFERIOR";K
4Ø CLS
5Ø FOR M = Ø TO 127
6Ø  SET(M,K)
7Ø NEXT M
8Ø RESET(X,Y - 47)
9Ø SET(X,Y)
1ØØ Y = Y + 1
11Ø IF Y < 48 THEN 13Ø
12Ø Y = Y - 48
13Ø IF<> K THEN 8Ø
999 GOTO 999
```

El punto parece chocar con la barrera y adherirse a ella.

Ahora, hagamos que el punto se origine en la mitad y que rebote en la parte superior e inferior.

```
1Ø CLS
2Ø FOR M = Ø TO 127
3Ø  SET(M,Ø)
4Ø  SET(M,47)
5Ø NEXT M
6Ø Y = 14
7Ø D = 1
8Ø RESET(64,Y+48-D)
9Ø SET (64,Y)
```

```
100 Y = Y + D
110 IF Y = 48 THEN 130
120 IF Y <> -1 THEN 80
130 Y = Y-2 * D
140 D = -D
150 GOTO 90
999 GOTO 999
```

El cambio de dirección del punto movil es debido a la línea 140 D = -D.
Observemos que hemos tomado precauciones para que no se borre la barrera. Para ésto, no sólo se mueve el punto dos pasos hacia atrás con la línea 130 (después de moverlo 1 hacia adelante con la línea 100), sino que también devolvemos el control del programa a la línea 90 donde se encuentra el SET, en lugar de mandarlo a la línea 80 donde está el RESET.

El siguiente programa es una muestra de las posibilidades de dibujo en la pantalla de video. Vamos a dibujar un hombrecillo.

Primero sus piernas :

```
10 CLS
20 X = 64
30 FOR K = 0 TO 7
40 SET(X + K, 40 + K)
50 SET(X - K, 40 + K)
60 NEXT K
999 GOTO 999
```

... y RUN.

Luego le añadimos el cuerpo y los brazos :

```
70 FOR K = 0 TO 5
80 SET(X-K,34+K)
```

```
9Ø SET(X,34+K)
1ØØ SET(X-K,34+K)
11Ø NEXT K
```

... y RUN.

y finalmente la cabeza :

```
12Ø SET(X,32)
13Ø SET(X+1,33)
14Ø SET(X-1,33)
```

... y RUN.

Ahora vamos a hacer que se mueva hacia la derecha :

```
45 RESET(X+K-1,4Ø+K)
55 RESET(X-K-1,4Ø+K)
85 RESET(X+K-1,34+K)
95 RESET(X-1,34+K)
1Ø5 RESET(X-K-1,34+K)
125 RESET (X-1,32)
135 RESET (X,33)
145 RESET(X-2,33)
15Ø X = X + 1
16Ø GOTO 3Ø
```

... y RUN.

Indudablemente todos los programas escritos en este capítulo son sólo una muestra de las posibilidades de las declaraciones SET y RESET. Intercambiando las citadas declaraciones en los programas que acabamos de ver, obtendremos figuras oscuras sobre fondo blanco.

Podemos hacer alguna prueba para comprobarlo. De cualquier forma pensamos que tenemos elementos suficientes para elaborar nuestros propios programas gráficos y visualizados en la pantalla del TRS-80.

Material Aprendido en el Capítulo 20

DECLARACIONES

SET

RESET

SOLUCION A LOS EJERCICIOS PROPUESTOS EN EL CAPITULO 20

RESPUESTA TIPO AL EJERCICIO 20-1 :

```
1Ø INPUT "BLOQUE HORIZONTAL INICIAL (Ø a 127)";H
2Ø INPUT "BLOQUE HORIZONTAL FINAL (Ø a 127)";I
3Ø INPUT "BLOQUE VERTICAL INICIAL (Ø a 47)";V
4Ø INPUT "BLOQUE VERTICAL FINAL (Ø a 47)";W
5Ø CLS
6Ø FOR X = H TO I
7Ø  FOR Y= V TO W
8Ø   SET(X,Y)
9Ø  NEXT Y
1ØØ NEXT X
999 GOTO 999
```

RESPUESTA TIPO PARA EL EJERCICIO 20-2 :

Se cambian las siguientes líneas. El resto es igual.

```
5Ø FOR L = X TO X + K * 2 + 1
7Ø SET(L, Y+K+1)
9Ø FOR M = Y TO Y+K+1
11Ø SET(X+K * 2+1,M)
```

RESPUESTA TIPO PARA EL EJERCICIO 20-3:

A. PARA MOVER EL PUNTO HACIA ARRIBA

```
1Ø INPUT "PUNTO HORIZONTAL INICIAL (Ø a 127)";X
2Ø INPUT "PUNTO VERTICAL INICIAL (Ø a 47)";Y
3Ø CLS
4Ø RESET(X,Y+1)
5Ø SET(X,Y)
6Ø Y = Y - 1
```

```
7Ø IF Y >=Ø THEN 4Ø
8Ø Y = Y+48
9Ø GOTO 4Ø
99 GOTO 99
```

ARA MOVER EL PUNTO HACIA LA IZQUIERDA

```
1Ø INPUT "PUNTO HORIZONTAL INICIAL (Ø a 127)";X
2Ø INPUT "PUNTO VERTICAL INICIAL (Ø a 47)";Y
3Ø CLS
4Ø RESET(X+1,Y)
5Ø SET(X,Y)
6Ø X = X - 1
7Ø IF X > =Ø THEN 4Ø
8Ø X = X+128
9Ø GOTO 4Ø
999 GOTO 999
```

CAPITULO 21

CONJUNTOS DE VARIABLES SUBINDICADAS (A VECES DENOMINADAS ARREGLOS)

Como se sabe, disponemos de 26 letras del alfabeto para ser usadas como variables. Hasta ahora, hemos utilizado sólo algunas en nuestros programas porque no hemos tenido necesidad de más. Sin embargo, a veces es necesario tener disponibles cientos o miles de variables para ser utilizadas como identificaciones de datos que se quieren almacenar y extraer fácilmente.

Para resolver este problema se utiliza el conjunto unidimensional de variables subindicadas ó "vector". Un vector nos permite representar una "serie de cosas", una "fila" ó "hilera". Supongamos que estamos hablando de una colección o hilera de 10 coches; todos ellos poseen diferentes placas (números de identificación) del 1 al 10 y queremos usar la computadora para que rápidamente nos dé el tamaño del motor cuando identifiquemos el coche por su número de placa. Imaginemos que los datos disponibles son los siguientes :

PLACA #	CILINDRADA DEL MOTOR (en pulgadas cúbicas)
1	3ØØ
2	2ØØ
3	5ØØ
4	3ØØ
5	2ØØ
6	3ØØ
7	4ØØ

8	4ØØ
9	3ØØ
1Ø	5ØØ

Ahora se le podía dar a cada coche, un nombre alfabético diferente, usando las variables de la A a la J pero, ¿qué pasaría si en lugar de 10 fueron 100 coches?

El NIVEL I del TRS-80 dispone de un sólo vector que se designa con la letra "A". No es igual a la variable alfabética "A" y no tiene el mismo sentido que la variable encadenada A$. Hablamos ahora del tercer tipo de variable A, que se reconocerá como una A(I). En nuestro ejemplo, representaremos los coches por las variables que van de A(1) hasta A(1Ø). La variable I que permite identificar a cada A(I) se denomina índice y la variable A(I) recibe el nombre de variable subindicada.

Almacenemos a continuación las distintas cilindradas de los motores de los coches, mediante una o más líneas de declaraciones DATA.

Escribamos :

```
1ØØ DATA 3ØØ,2ØØ,5ØØ,3ØØ,2ØØ
11Ø DATA 3ØØ,4ØØ,4ØØ,3ØØ,5ØØ
```

Podemos observar que los elementos del DATA aparecen en orden, de modo que la primera cilindrada de motor corresponda al primer coche, la segunda al segundo y así sucesivamente.

Nos hallamos ahora en condiciones de direccionar los citados elementos del DATA haciendo uso de vector.

La forma sencilla de generar un vector es la que se indica a continuación.

Escribamos :

```
50 FOR L = 1 TO 10
55 READ A(L)
60 NEXT L
```

... y RUN.

Lógicamente no sucede nada, porque todavía no hemos mandado imprimir los resultados, de cualquier forma READY ha tardado un poco de tiempo en aparecer lo que demuestra que el ordenador ha estado realizando algo.

Como puede apreciarse, se ha utilizado un bucle FOR -NEXT para leer los 10 elementos DATA que se almacenan en las variables comprendidas entre A(1) y A(10).

Para imprimir los valores de los elementos del vector, escribamos :

```
200 FOR N = 1 TO 10
210 PRINT A(N)
220 NEXT N...
```

y RUN.

Como puede apreciarse, se han leído los elementos del DATA dentro de un vector llamado A(L) pero se han impreso dentro de otro llamado A(N); esto no tiene consecuencias para la ejecución del programa, ya que, lo que importa es el número que representa la letra L y N que es el que identifica la posición de cada elemento dentro del arreglo. Así, se podía haber utilizado cualquiera de las 26 variables o incluso ecuaciones aritméticas dentro de los paréntesis si hubiera sido necesario.

Hay que recordar que sólo existe un vector cuyo nombre es "A" para el nivel I (BASIC I).

Ampliando el programa se puede escribir :

```
170 PRINT
180 PRINT "PLACA #", "CILINDRADA DEL MOTOR"
210 PRINT N, A(N)
```

... y RUN.

De esta manera se han colocado títulos al vector y tenemos el número de placa y de motor en dos columnas que poseen una correspondencia biunívoca.

Borremos a continuación las líneas 200,210,220 y escribamos :

```
10 IN. "DE QUE COCHE DESEA CONOCER LA CILINDRADA DEL MOTOR";W
210 PRINT W, A(W)
```

... y RUN.

Como puede apreciarse, éste podría ser el comienzo de un sencillo sistema de inventario.

Supongamos ahora que se desea conocer también el color de cada uno de los 10 coches dentro de una clave de colores que van desde el uno hasta el cuatro. Necesitaríamos entonces una tabla como la siguiente :

PLACA #	TAMAÑO DEL MOTOR	CLAVE DEL COLOR
1	300	3
2	200	1
3	500	4
4	300	3
5	200	2
6	300	4
7	400	3
8	400	2
9	300	1
10	500	3

La tabla anterior recibe el nombre de "matriz".
Una matriz es un conjunto de variables subindicadas con una dimensión superior a la unidad.
El primer conjunto estudiado tenía una dimensión unidad, ya que, poseía una sola fila ó columna (tenía la dimensión del "largo"). El conjunto de la tabla tiene dos dimensiones correspondientes a sus filas y columnas, sin contar el nº de placa (dimensión 2 : "largo" x "ancho").

Para trabajar con esta matriz de 10 filas x 2 columnas, nos interesa en primer lugar conocer la memoria libre de que disponemos. Para ello, suponiéndo que no se ha alterado el programa introducido, escribamos :

```
PRINT MEM
```

se debe obtener una respuesta de aproximadamente ...

```
337Ø
```

Existen 337Ø celdas bytes de memoria sin utilizar. Usando la modalidad de calculadora, escribamos :

```
PRINT 337Ø/4-1
```

y obtendremos :

```
841.5
```

Hemos encontrado así el espacio disponible para los elementos de nuestra matriz, ya que cada elemento ocupa cuatro bytes de memoria.

Como existe todavía mucha memoria libre, vamos a asignar las posiciones 1Ø1 a la 11Ø para que almacenen la clave del color, usando una declaración DATA. Escribamos :

```
3ØØ DATA 3,1,4,3,2,4,3,2,1,3
8Ø FOR S = 1Ø TO 11Ø
85 READ A(S)
9Ø NEXT S
```

Los números de los elementos del arreglo del 11 al 1ØØ, no se utilizan, así como tampoco los comprendidos entre el 111 y el final de la memoria, ya que no se les ha asignado valores.

Para imprimir toda esta información, cambiemos las líneas :

```
1Ø IN. "DE QUE COCHE DESEA CONOCER LA CILINDRADA DEL MOTOR
   Y EL COLOR?"; W
18Ø P. "PLACA #", "CILINDRADA DEL MOTOR", "CLAVE DEL COLOR"
21Ø PRINT W, A(W), A(W + 1ØØ)
```

... y RUN.

Esta es la forma de manejar tablas gráficas o listas de inventario. Ahora se podía buscar, igual que se hizo antes, un coche que tenga cierta cilindrada y cierto color.

Como podemos observar nos hemos visto obligados a introducir los elementos de una tabla bidimensional en un vector unidimensional. En el BASIC NIVEL 2, en cambio, dispondremos de matrices pluridimensionales.

EJERCICIO 21-1 : Suponiendo que el inventario de 10 coches, incluye tres modelos diferentes con las claves 1Ø, 2Ø y 3Ø como se indica a continuación ...

PLACA #	MODELO
1	2Ø
2	2Ø
3	1Ø
4	2Ø
5	3Ø
6	2Ø
7	3Ø
8	1Ø
9	2Ø
1Ø	2Ø

La tabla anterior recibe el nombre de "matriz".
Una matriz es un conjunto de variables subindicadas con una dimensión superior a la unidad.
El primer conjunto estudiado tenía una dimensión unidad, ya que, poseía una sola fila ó columna (tenía la dimensión del "largo"). El conjunto de la tabla tiene dos dimensiones correspondientes a sus filas y columnas, sin contar el nº de placa (dimensión 2 : "largo" x "ancho").

Para trabajar con esta matriz de 10 filas x 2 columnas, nos interesa en primer lugar conocer la memoria libre de que disponemos. Para ello, suponiéndo que no se ha alterado el programa introducido, escribamos :

```
PRINT MEM
```

se debe obtener una respuesta de aproximadamente ...

```
337Ø
```

Existen 337Ø celdas bytes de memoria sin utilizar. Usando la modalidad de calculadora, escribamos :

```
PRINT 337Ø/4-1
```

y obtendremos :

```
841.5
```

Hemos encontrado así el espacio disponible para los elementos de nuestra matriz, ya que cada elemento ocupa cuatro bytes de memoria.

Como existe todavía mucha memoria libre, vamos a asignar las posiciones 1Ø1 a la 11Ø para que almacenen la clave del color, usando una declaración DATA. Escribamos :

```
3ØØ DATA 3,1,4,3,2,4,3,2,1,3
8Ø FOR S = 1Ø TO 11Ø
85 READ A(S)
9Ø NEXT S
```

Los números de los elementos del arreglo del 11 al 1ØØ, no se utilizan, así como tampoco los comprendidos entre el 111 y el final de la memoria, ya que no se les ha asignado valores.

Para imprimir toda esta información, cambiemos las líneas :

```
1Ø IN. "DE QUE COCHE DESEA CONOCER LA CILINDRADA DEL MOTOR
   Y EL COLOR?"; W
18Ø P. "PLACA #", "CILINDRADA DEL MOTOR", "CLAVE DEL COLOR"
21Ø PRINT W, A(W), A(W + 1ØØ)
```

... y RUN.

Esta es la forma de manejar tablas gráficas o listas de inventario. Ahora se podía buscar, igual que se hizo antes, un coche que tenga cierta cilindrada y cierto color.

Como podemos observar nos hemos visto obligados a introducir los elementos de una tabla bidimensional en un vector unidimensional. En el BASIC NIVEL 2, en cambio, dispondremos de matrices pluridimensionales.

EJERCICIO 21-1 : Suponiendo que el inventario de 10 coches, incluye tres modelos diferentes con las claves 1Ø, 2Ø y 3Ø como se indica a continuación ...

PLACA #	MODELO
1	2Ø
2	2Ø
3	1Ø
4	2Ø
5	3Ø
6	2Ø
7	3Ø
8	1Ø
9	2Ø
1Ø	2Ø

Modificar el programa, para que imprima la información del modelo del coche y la cilindrada, cuando se identifique un coche por su número de placa.

APLICACION DE UN CONJUNTO UNIDIMENSIONAL DE VARIABLES SUBINDICADAS (VECTOR)

En el ejemplo presente, se pretende generar una baraja de cartas mediante un vector, de manera que se escojan elementos del vector utilizando un generador de números al azar. A medida que se "roba" cada carta, se fija el elemento correspondiente a la carta del vector igual a cero, y se comprueba cada carta robada para ver si es o no cero. De esta manera se puede completar la baraja sin repeticiones de cartas.

Se va a escribir paso a paso un programa que genera al azar las 52 cartas numeradas del 1 al 52, e imprime los números de las cartas sobre la pantalla conforme son generadas. No debe ser generada ninguna carta más de una vez. Cuando hayan sido generadas todas las cartas, se imprimirá "FIN DE LA BARAJA".

Se recomienda adelantarse a los pasos sucesivos del ejercicio e ir comparando las soluciones con las del texto.

Paso 1º : Generar las 52 cartas completas dentro de un vector, solución :

```
3Ø FOR C = 1 TO 52:READ A(C):NEXT C
5Ø DATA 1,2,3,4,5,6,7,8,9,1Ø,11,12,13,14,15,16,17,18,19,2Ø
55 DATA 21,22,23,24,25,26,27,28,29,3Ø,31,32,33,34,35,36
6Ø DATA 37,38,39,4Ø,41,42,43,44,45,46,47,48,49,5Ø,51,52
```

Paso 2º : Barajar 52 cartas al azar, imprimiendo sus valores. Solución :

```
9Ø FOR N = 1 TO 52
1ØØ V = RND(52)
11Ø PRINT A(V);
12Ø NEXT N
```

Como puede verse, se imprimen 52 valores de cartas en la pantalla, pero el mismo número aparece más de una vez y existen otros en cambio que no aparecen.

Paso 3º : Cuando se extrae una carta, hagamos que la variable subindicada del vector correspondiente a ese índice sea igual a cero. Cuando se hayan extraido las 52 cartas, debe aparecer en la pantalla : FIN DE LA BARAJA.

```
9Ø P = 52
1Ø5 IF A(V) = Ø G. 1ØØ
12Ø LET A(V) = Ø
13Ø P = P - 1
14Ø IF P<>Ø GOTO 1ØØ
15Ø PRINT "FIN DE LA BARAJA"
```

La línea 12Ø fija el valor de la variable A(V) igual a cero sólo si la línea 1Ø5 encuentra que no es igual a cero, permitiendo que el programa continúe.

A continuación observamos que :

1. Se imprime el valor de la carta (línea 11Ø).
2. El número almacenado en esa variable subindicada se iguala a Ø (línea 12Ø).
3. La línea 13Ø cuenta en forma decreciente el número de cartas que quedan por extraer. La línea 9Ø inicializa el número de cartas a 52.
4. La línea 14Ø comprueba el número de cartas extraídas. Cuando este número alcanza el valor 52, se imprime " FIN DE LA BARAJA" en la línea 15Ø.

Una vez ejecutado el programa, se puede efectuar una prueba que demuestre que todas las cartas han sido barajadas. Escribir :

```
200 FOR T = 1 TO 52
210 PRINT A(T);
220 NEXT T
```

... y RUN. La pantalla debe mostrar todas las variables subindicadas puestas a cero. Para verlo más claramente, cambiemos la línea 90 por :

```
90 P = 50
```

... y RUN.

Podría decirse que la microcomputadora se ha sacado dos cartas de la manga.

Inicialicemos de nuevo P a 52; eliminemos las líneas de prueba 200, 210 y 220, para dejar disponible una interesante subrutina capaz de extraer de forma aleatoria las cartas de una baraja, que puede ser guardada en cinta para posteriores usos.

Pregunta : ¿Por qué se hace lenta la impresión de los números de las cartas hasta casi detenerse cuando se están generando las últimas cartas ?

Respuesta : El generador de números al azar debe seguir generando números índices de variables subindicadas, hasta que llegue a aquél cuya variable subindicada no haya sido ajustada a cero. Casi al final de la baraja, la mayoría de los elementos han sido igualados a cero. El generador de números al azar debe seguir generando números tan rápido como pueda para encontrar algún número "útil". Revisar una vez más los números de las cartas extraidas de la baraja. No existe duplicidad.

Por último, se puede utilizar la variable subindicada A(I), en el improbable caso de que se empleen las 26 variables alfabéticas. Sin embargo, hay que tener en cuenta que cada una de ellas requiere cuatro veces el espacio de memoria necesario para una letra del alfabeto. Borremos la memoria y escribamos :

```
10 A(870) = 3
20 A(871) = 4
30 Z = A(870) + A(871)
40 PRINT A(870),A(871),Z
```

... y RUN.

Por supuesto, las respuestas son :

```
3                    4                    12
```

Ahora probemos :

```
10 A(1000) = 3
```

... y RUN.

El mensaje de error SORRY se debe a que el número mayor de índice de variable que puede utilizarse, es el 875 (con memoria de 4K por supuesto).

Material Aprendido en el Capítulo 21

VARIOS

Conjuntos de variables subindicadas.

SOLUCION A LOS EJERCICIOS PROPUESTOS EN EL CAPITULO 21

RESPUESTA TIPO PARA EL EJERCICIO 21-1 :

Hemos de agregar o cambiar las siguientes líneas :

```
1Ø IN. "DE QUE COCHE QUIERE CONOCER LA CILINDRADA DEL
   MOTOR, EL COLOR Y EL MODELO; W
13Ø FOR B = 2Ø1 TO 21Ø
135 READ A(B)
14Ø NEXT B


18Ø P. "PLACAS #", "TAMAÑO DEL MOTOR", "CODIGO DEL COLOR",
    "MODELO"

21Ø P. W,A(W),A(W+1ØØ),A(W+2ØØ)
4ØØ DATA 2Ø,2Ø,1Ø,2Ø,3Ø,2Ø,3Ø,1Ø,2Ø,2Ø
```

CAPITULO 22

GRAFICAS AVANZADAS

Volviendo al programa del punto rebotante supongamos que pretendemos visualizar, de alguna forma el instante en que el punto choca con la barrera, por ejemplo imprimiendo PING.

Conocemos varias formas de declaración PRINT pero ninguna se ajusta al objetivo perseguido.

Una nueva declaración, de especial utilidad en las gráficas es PRINT AT. Además en este capítulo estudiaremos otra declaración, que aparece bajo la forma POINT(X,Y) y que permite detectar si cualquiera de los 6144 bloques gráficos, está encendido o apagado. Estas dos nuevas declaraciones sumadas a las ya conocidas aumentan las posibilidades de trabajo con gráficas.

LA DECLARACION PRINT AT

La declaración PRINT AT # # # permite comenzar la escritura de un texto, en el punto indicado # # #. Pero ¿dónde está este punto?. Escribamos :

```
10 CLS
50 PRINT AT 200,"HOLA 200, DONDE QUIERA QUE ESTES"
```

... y RUN.

Volviendo al capítulo 20 en la esquina superior izquierda de la hoja de trabajo del display de video, observaremos en letra muy pequeña las palabras "PRINT AT".

También puede verse una flecha que apunta hacia una serie de números. Además aparece una "x" minúscula, que se refiere evidentemente a las direcciones de la fila X y una "y" minúscula que hace referencia a las direcciones de la columna Y. Si se observa detenidamente, se pueden ver los 64 números del "TAB" empezando por el cero.

Los números PRINT AT empiezan en Ø y van hasta 63 - en la primera línea. Luego continúan sobre la segunda línea a partir del número ≠ 64 hasta el 127. La tercera línea comienza con el número 128, etc. Las divisiones PRINT AT son en realidad las mismas que para el TAB excepto que PRINT AT no se inicia nuevamente con Ø en la segunda línea. PRINT AT continúa hasta la posición 1Ø23.

La declaración PRINT AT, puede direccionar a voluntad cualquiera de las 1Ø23 posiciones de impresión (no confundir con bloques luminosos). Se necesitan puntos y comas en la última línea (posiciones de la 96Ø a la 1Ø23), para evitar el retorno del carro y evitar así que se mueva hacia arriba la "página de la pantalla".

A continuación veamos una aplicación de la declaración PRINT AT, repasando además el concepto de bucles anidados. Se trata de generar un reloj de 24 horas. Escribamos :

```
1Ø CLS
2Ø PRINT AT 4Ø7, "H  M  S"
3Ø  FOR H = Ø TO 23
4Ø   FOR M = Ø TO 59
5Ø    FOR S = Ø TO 59
6Ø     PRINT AT 47Ø,H; ":";M; ":";S
7Ø      FOR N = 1 TO 5ØØ:NEXT N
8Ø    NEXT S
9Ø   NEXT M
1ØØ NEXT H
```


... Y RUN

Por supuesto, la exactitud de este reloj depende de la precisión de la calibración. La TRS-80 NIVEL I ejecuta algo así como 500 bucles FOR-NEXT sencillos por segundo cuando la escritura del bucle es tal como la indicada en la línea 7Ø (línea declarativa múltiple). La manera de realizar ajustes en el reloj, será entonces incrementar o decrementar el valor 5ØØ.

De cualquier manera no hay que perder de vista que el objetivo es estudiar la declaración PRINT AT; por lo tanto vamos a dirigirnos a la hoja de trabajo para la pantalla de video donde se aprecia que la dirección 47Ø cae inmediatamente debajo de la 4Ø6. De esta manera se indica que los primeros números son las horas, los segundos los minutos y los terceros los segundos.

Otro ejemplo que muestra el manejo de la declaración PRINT AT, es el siguiente programa, que sobrepone un formato de gráfica sobre la pantalla, que puede resultar útil para muchos usos. Vale la pena escribirlo y estudiar sus pasos.

```
1Ø CLS
2Ø K = 9ØØ
3Ø F.X = 1 TO 59
4Ø P. ATK + X,".";
5Ø N.X:K = 964
6Ø FOR Y = Ø TO 13
7Ø P. ATY*64 + 5,".";
8Ø NEXT Y
1ØØ P. AT2Ø,"TITULO DE LA GRAFICA"
15Ø F. N = Ø TO 14
2ØØ    P. ATN*64, 14 - N;
25Ø N.N
```

```
300 F.X = 0 TO 5
310   P.AT K + 10* X,X;
320 N.X
400 F.X = 6 TO 56 STEP 10
410   P.AT K + K, ":";
420 N.X
999 G.999
```

NOTA :

Podemos observar que en este programa aparece un punto y coma al final de las declaraciones PRINT AT. Ello es debido a que al ser impresa la última línea de la pantalla, se activaría el contaje consiguiéndose suprimir el retorno del carro mediante el punto y coma.

EJERCICIO 22-1 : Referirse al programa de la pelota rebotante :

```
10 CLS
20 FOR M = 0 TO 127
30  SET(M,0)
40  SET(M,47)
50 NEXT M
60 Y = 14
70 D = 1
80 RESET (64,Y + 48 -D)
90 SET(64,Y)
100 Y = Y + D
110 IF Y = 48 THEN 130
120 IF Y <> -1 THEN 80
130 Y = Y - 2 *D
140 D = -D
150 GOTO 90
999 GOTO 999
```

Usar declaraciones PRINT AT, para que la palabra "PING" aparezca cerca de la pelota cada vez que rebote, ya sea, en la barrera superior o inferior.

LA DECLARACION POINT

La declaración POINT(X,Y) permite conocer la situación luminosa, de un punto cualquiera de la pantalla. Si el punto está iluminado se obtendrá un 1 y en caso contrario un Ø.

La declaración POINT trabaja incluso en la modalidad de calculadora. Para comprobarlo escribamos :

```
PRINT POINT (3Ø,3Ø)
```

como la dirección (3Ø,3Ø) no se había iluminado con anterioridad, la respuesta ha sido Ø. La instrucción puede abreviarse :

```
P.P. (3Ø,3Ø)
```

se obtiene la misma respuesta Ø.

Elaboremos a continuación un programa que nos permita seguir el rastro del punto luminoso.

```
1Ø S = 1:N = Ø
2Ø IN. "DESEA ENCENDER EL BLOQUE (S/N)?";Q
3Ø CLS
4Ø IF Q = Ø GOTO 8Ø
5Ø SET(75,2Ø)
6Ø GOTO 1ØØ
8Ø RESET(75,2Ø)
1ØØ IF POINT(75,2Ø) = 1 P.AT 2ØØ,X;Y, "ESTA ENCENDIDO"
2ØØ IF POINT(75,2Ø) = Ø P.AT 2ØØ,X;Y, "ESTA APAGADO"
999 G. 999
```

... y RUN.

A la pregunta ¿desea encender el bloque (S/N)?, se puede responder con una S ó una N, también con SI ó NO ó cualquier palabra que empiece por S ó por N, ya que sólo estas variables tienen asignados los valores 1 y Ø.

Se acaba de realizar una especie de encuesta de la situación en que se halla un punto determinado de la pantalla. Mediante dos bucles anidados es posible rastrear toda la pantalla observando la situación de cada punto. Escribamos y ejecutemos el siguiente programa, observando su funcionamiento y deteniendo la ejecución temporalmente con la tecla (↑) :

```
1Ø REM *DEMOSTRACION DE LA DECLARACION GRAFICA POINT*
2Ø P = 15:L = 119
3Ø CLS
4Ø P.AT5, "ESTA ES UNA DEMOSTRACION DE LA DECLARACION
   POINT--";
5Ø P.AT56,"X          Y";
1ØØ F.I = 1 TO P:SET(RND(113),RND(45) + 2):N.I
11Ø F.X = Ø TO111:F.Y = Ø TO 47
12Ø   IF POINT(X,Y) = Ø GOTO 16Ø
13Ø   P.AT L,X;:P.AT L + 4, Y;
14Ø   L = L + 64
15Ø   G. 17Ø
16Ø   SET(X,Y):RESET(X,Y)
17Ø N.Y:N.X
18Ø P.AT5 "LAS COORDENADAS DE LOS BLOQUES DE LAS GRAFICAS
    SON >>-- >> ";
19Ø P.ATØ;
2ØØ G.2ØØ
```

Explicación del programa :

La línea 1Ø es sólo una nota de identificación del programa.

La línea 2Ø fija P a 15, (el número de "blancos" a ser puestos al azar sobre la pantalla por el bucle FOR-NEXT y el generador de números al azar de la línea 1ØØ). L se fija a 119, y es el punto de partida para imprimir las coordenadas y sus títulos en las líneas 130 y 140.

La línea 3Ø borra la pantalla.

Las líneas 4Ø y 5Ø usan PRINT AT para imprimir los títulos.

La línea 1ØØ genera 15 direcciones y sitúa (set) 15 bloques luminosos al azar en la pantalla.

La línea 11Ø utiliza dos bucles FOR-NEXT anidadas para que pueda seguirse el rastro de cada uno de los puntos gráficos de la pantalla.

La línea 12Ø comprueba si el punto que se está direccionando está apagado. Si es así, se omite la impresión de la dirección (líneas 13Ø y 14Ø).

La línea 13Ø imprime los valores de X y de Y si consiguen pasar la prueba impuesta por la declaración POINT en la línea 12Ø, lo cual significa que el punto que se estaba probando no estaba apagado.

La línea 14Ø incrementa la dirección de la PRINT AT para que la línea 13Ø imprima las coordenadas adecuadas en la siguiente ocasión en que haya de hacerlo. Agregando 64, la siguiente impresión se alineará directamente debajo del título y los números actuales. (Ver la hoja de trabajo para la pantalla de video).

La línea 15Ø es una línea de salto sobre un reset. Sin esta línea, los bloques de luz distribuidos al azar serían borrados de la pantalla cuando pasa por ellos el bloque luminoso móvil. Se puede efectuar la comprobación eliminando la línea.

La línea 16Ø es la que permite al bloque luminoso móvil que se ilumine y apague.
Cuando se llega a un bloque iluminado, el programa se salta esta línea.

La línea 17Ø sólamente cierra el bloque FOR-NEXT que se inició en la línea 11Ø.

La línea 18Ø reemplaza el título que fue borrado por el proceso de rastreo del POINT.

La línea 19Ø simplemente mueve el cursor a la posición Ø, o lo que es lo mismo, lo manda a la parte superior izquierda de la pantalla.

La línea 2ØØ es un bucle sin fin, que mantiene la pantalla libre del READY y del apuntador.

Si nos fijamos, el bloque móvil elimina el título original de la gráfica. También puede observarse que se necesitan dos pasadas para igualar el ancho de un carácter impreso. Se podía haber protegido el título, como se protegieron los bloques, ó reducir la longitud vertical del rastreo.

Material Aprendido en el Capítulo 22

FUNCIONES GRAFICAS

POINT (X,Y)
PRINT AT

SOLUCION A LOS EJERCICIOS PROPUESTOS EN EL CAPITULO 22

RESPUESTA TIPO PARA EL EJERCICO 22-1 :

Insertar las siguientes líneas :

```
105 IF Y = 46 THEN 180

115 IF Y = 1 THEN 180

150 PRINT AT Y*64 + 32, "  "

160 G.90

180 PRINT AT Y *64 + 32, "PING"

190 G.90
```

Observese que la línea 180 imprime el "PING" y la línea 150 lo hace desaparecer imprimiendo espacios en su lugar.

MINIORDENADOR

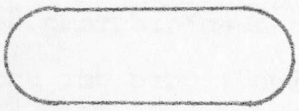
PARTE 2

CAPITULO 23

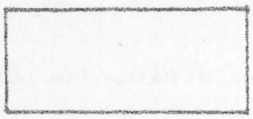
DIAGRAMAS DE FLUJO

Hasta ahora los programas que se han escrito han sido sencillos. Supongamos que queremos escribir un programa para jugar al ajedrez o a las cartas, que evalue alternativas de inversión complicadas, que lleve la contabilidad de un pequeño negocio o que realice cálculos de esfuerzos en un edificio. ¿Cómo empezaríamos a escribir un programa complicado como cualquiera de estos?

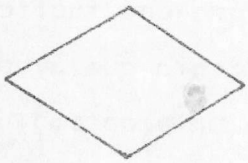
Una manera de elaborar un programa, es expresar gráficamente su lógica interna. El dibujo que utilizan los programadores se llama "diagrama de flujo". Los diagramas de flujo se utilizan tanto que se han establecido símbolos convencionales para unificar criterios entre programadores. Existen muchos símbolos especializados en uso, pero examinaremos sólo los más comunes.



INICIO O FIN



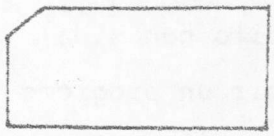
BLOQUE DE PROCESO
(Incluye procesos que la computadora realiza sin tomar decisiones).



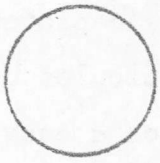
BLOQUE DE CONDICION
(se ramifica en diferentes direcciones dependiendo de la decisión que tome)

Cada punto de decisión formula una pregunta tal como ¿ES A MAYOR QUE B? o cualquier otra. Las diferentes ramas se marcan con un SI ó NO.

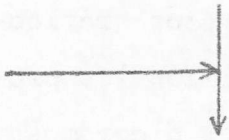
BLOQUE DE SALIDA DE DATOS
(Contiene instrucciones de impresión de datos en pantalla, impresora, cinta, etc.)



BLOQUE DE ENTRADA DE DATOS
(Contiene instrucciones de entrada de datos por teclado, cinta, etcc...)



CONTINUACION
(Contiene un número que corresponde a la continuación en otra página del programa).



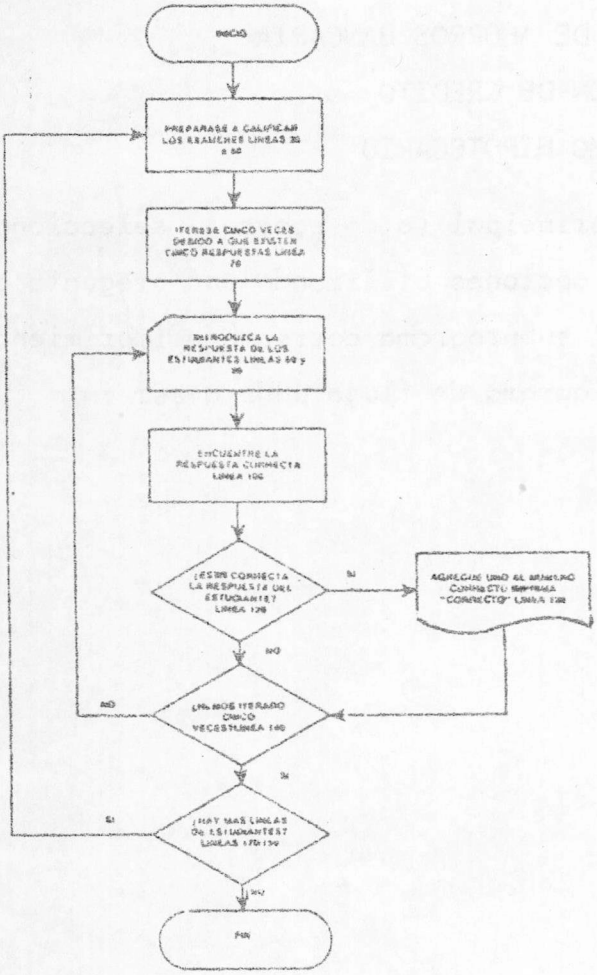
FLECHAS DE CONEXION
(Sirven para la interconexión de los bloques).

Los diagramas de flujo son de mucha utilidad en el diseño de programas cuando se dibujan ateniéndose a criterios de sencillez y claridad. Un diagrama de flujo desordenado es difícil de leer y generalmente no será de más utilidad que un listado de programa ordinario. Un buen diagrama de flujo es también útil para la "documentación", para dar una imagen rápida de cómo trabaja el programa cuando ya no se le recuerda.

No existen reglas estrictas sobre la elaboración de los diagramas de flujo (como tampoco existen para la realización de programas); cada programador utiliza una táctica particular. El diagrama de flujo es la herramienta que ayuda a sintetizar ideas con respecto a un programa. Un programa, por muy complicado que sea, se puede estructurar partiendo de un diagrama de flujo muy simple en el cual, cada bloque puede englobar otros diagramas de flujo, que se irán obteniendo de forma paulatina y sucesiva.

Veamos algunos ejemplos. Supongamos que queremos calificar una prueba de cinco preguntas comparando cada una de las respuestas de los estudiantes con la respuesta correcta. Pondremos las respuestas correctas en una declaración DATA del programa, introduciendo las respuestas de los estudiantes a través del teclado. El programa tiene que comparar e imprimir el % de respuestas correctas. Este procedimiento debe repetirse hasta que sean calificadas todas las pruebas de los estudiantes.

El diagrama de flujo podía ser :



Este diagrama de flujo tiene tres bloques (rombos) de decisión. En el primero, la computadora decide si una respuesta es correcta. En la segunda, la computadora determina si han sido calificadas todas las respuestas del exámen de un estudiante (este bloque sería innecesario utilizando un bucle FOR-NEXT).

La tercera, finaliza la ejecución cuando se hayan calificado todas las pruebas.

EJERCICIO 23-1 : Utilizando el diagrama de flujo que acabamos de ver, escribir un programa que califique una prueba que contenga cinco respuestas.

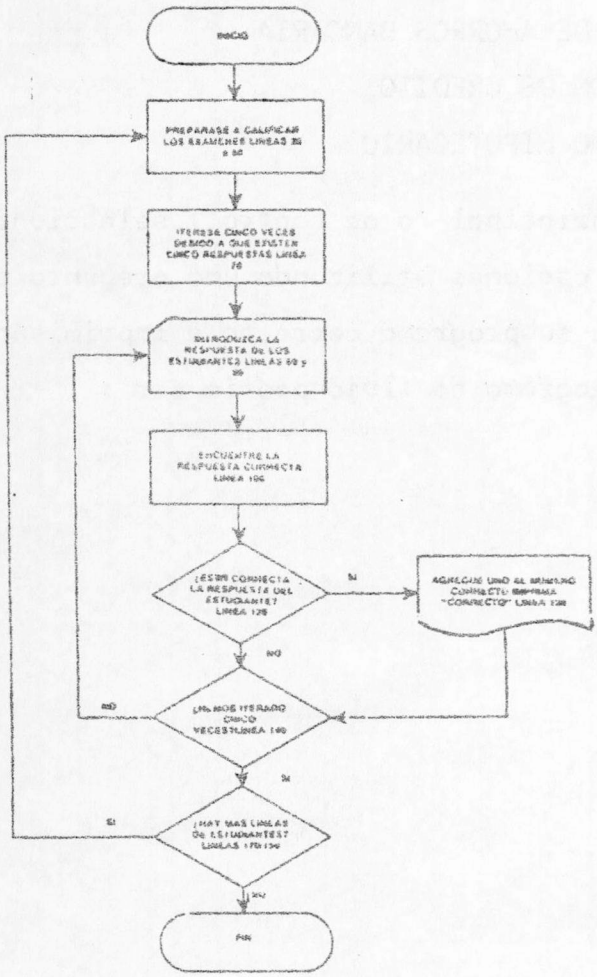
Otro ejemplo podría ser el cálculo de las ganancias de varias inversiones. Supongamos que tenemos las siguientes opciones :

1 CERTIFICADO DE DEPOSITO
2 CUENTA DE AHORROS BANCARIA
3 LA UNION DE CREDITO
4 PRESTAMO HIPOTECARIO

El programa principal (o de control) seleccionará una de estas cuatro opciones utilizando una pregunta de entrada, ejecutando el subprograma correcto e imprimiendo la respuesta. El diagrama de flujo podría ser :

Veamos algunos ejemplos. Supongamos que queremos calificar una prueba de cinco preguntas comparando cada una de las respuestas de los estudiantes con la respuesta correcta. Pondremos las respuestas correctas en una declaración DATA del programa, introduciendo las respuestas de los estudiantes a través del teclado. El programa tiene que comparar e imprimir el % de respuestas correctas. Este procedimiento debe repetirse hasta que sean calificadas todas las pruebas de los estudiantes.

El diagrama de flujo podía ser :



Este diagrama de flujo tiene tres bloques (rombos) de decisión. En el primero, la computadora decide si una respuesta es correcta. En la segunda, la computadora determina si han sido calificadas todas las respuestas del exámen de un estudiante (este bloque sería innecesario utilizando un bucle FOR-NEXT).

La tercera, finaliza la ejecución cuando se hayan calificado todas las pruebas.

EJERCICIO 23-1 : Utilizando el diagrama de flujo que acabamos de ver, escribir un programa que califique una prueba que contenga cinco respuestas.

Otro ejemplo podría ser el cálculo de las ganancias de varias inversiones. Supongamos que tenemos las siguientes opciones :

1 CERTIFICADO DE DEPOSITO
2 CUENTA DE AHORROS BANCARIA
3 LA UNION DE CREDITO
4 PRESTAMO HIPOTECARIO

El programa principal (o de control) seleccionará una de estas cuatro opciones utilizando una pregunta de entrada, ejecutando el subprograma correcto e imprimiendo la respuesta. El diagrama de flujo podría ser :

Se podían también sacar diagramas por separado de algunos de los módulos. El programa de certificado de depósito, por ejemplo, tendría que incluir la tasa de interés, el valor del depósito y el número de años.

EJERCICIO 23-2 : Escribir un programa basándose en el diagrama de flujo anterior de manera que calcule las ganancias en cuentas de ahorro bancarias pagando intereses simples. El resto de subprogramas (CERTIFICADO DE DEPOSITO, UNION DE CREDITO y PRESTAMO HIPOTECARIO), no es necesario escribirlos, y se simulan con un retardo de 4 segundos, después del cual se comienza introduciendo una nueva opción. Para aclarar mejor este punto diremos que las flechas que salen de estos módulos, en lugar de dirigirse al módulo "IMPRIME LAS GANANCIAS SOBRE LA INVERSION" se dirigen al de "INTRODUZCA LA OPCION".

EJERCICIO 23-3 : Escoger un programa de algún capítulo anterior y dibujar su diagrama de flujo.

Material aprendido en el Capítulo 23

VARIOS

Diagramas de Flujo

SOLUCION A LOS EJERCICIOS PROPUESTOS EN EL CAPITULO 23 --

RESPUESTA TIPO PARA EL EJERCICIO 23-1 :

```
10 REM * EVALUADOR DE EXAMENES *
20 CLS
30 P. "ESTE ES UN PROGRAMA EVALUADOR DE EXAMENES"
40 P. "ESCRIBA LAS CINCO RESPUESTAS DEL ESTUDIANTE
   SEGUN SE LE VAYAN SOLICITANDO"
50 RESTORE
60 N = 0
70 FOR I = 1 TO 5
80 PRINT "RESPUESTA NUMERO";I;
90 INPUT A
100 READ B
110 PRINT A,B;
120 IF A = B THEN PRINT "CORRECTO":N = N + 1
130 PRINT
140 NEXT I
150 PRINT N; "CORRECTAS LAS CINCO";
160 PRINT N/5 * 100; "%"
170 P. "¿HAY MAS EXAMENES POR CALIFICAR?";
180 IN. "  1= SI, 2 = NO";Z
190 IF Z = 1 GOTO 50
200 DATA 65,23,17,56,39
```

RESPUESTA TIPO PARA EL EJERCICIO 23-2 :

```
10 REM * RESPUESTA TIPO 23-2 *
100 CLS
110 P.:P
120 P. "ESCRIBA EL NUMERO DE ALGUNA DE LAS SIGUIENTES IN-
    VERSIONES"
130 P.
```

```
14Ø P. "  1 - CERTIFICADO DE DEPOSITO"
15Ø P. "  2 - CUENTA DE AHORROS BANCARIA"
16Ø P. "  3 - UNION DE CREDITO"
17Ø P. "  4 - PRESTAMO HIPOTECARIO"
18Ø P.:IN. "INVERSION";F
19Ø ON F GOTO 1ØØØ, 2ØØØ, 3ØØØ, 4ØØØ
2ØØ GOTO 1ØØ:REM SE UTILIZARA SI EL NUMERO NO ESTA
    ENTRE 1 Y 4
1ØØØ REM *EL PROGRAMA DE CERTIFICADO DE DEPOSITO VA AQUI *
1Ø1Ø P. " TIENE QUE ESCRIBIRSE EL PROGRAMA DE C.D."
1Ø2Ø GOS. 1ØØØØ:G.1ØØ
2ØØØ REM *PROGRAMA DE CUENTA DE AHORRO BANCARIA *
2Ø1Ø CLS:P.:P "ESTA RUTINA CALCULA EL INTERES SIMPLE
    SOBRE"
2Ø2Ø P."LA CANTIDAD MANTENIDA EN DEPOSITO DURANTE UN
    PERIODO ESPECIFICADO"
2Ø3Ø P. "UTILIZANDO UN PORCENTAJE DE INTERES ESPECIFICADO":
    P.
2Ø4Ø P.:IN. "CUAL ES EL DEPOSITO (EN DOLARES)";P
2Ø5Ø IN. "CUANTO TIEMPO LO GUARDARA (EN DIAS)";D
2Ø6Ø IN. "QUE TASA DE INTERES ESPERA (EN %)";R
2Ø7Ø CLS:P.:P.:P  "PARA UN CAPITAL INICIAL DE $";P;
    "A UNA"
2Ø8Ø P. "TASA DE";R; "% POR";D; "DIAS, EL INTERES"
2Ø9Ø P. "RESULTO SER";
21ØØ REM *INTERES =(%/YR)/(DIAS/YR) * DIAS* PRINCIPIO*
22ØØ I = R/1ØØ/365 * D *P
23ØØ P.:P. "    "; "$";I
24ØØ END
3ØØØ REM *EL PROGRAMA DE LA UNION DE CREDITO VA AQUI*
3Ø1Ø P. "EL PROGRAMA DE LA U.C. NECESITA ESCRIBIRSE"
3Ø2Ø GOS. 1ØØØØ:G.1ØØ
4ØØØ REM* EL PROGRAMA DE PRESTAMO HIPOTECARIO VA AQUI *
4Ø1Ø P.  "EL PROGRAMA DE P.H. NECESITA ESCRIBIRSE AQUI"
4Ø2Ø GOS.1ØØØØ:G.1ØØ
1ØØØØ F. I = 1 TO 2ØØØ:N.I:RET.
```

CAPITULO 24

LOGICA Y/O

En matemáticas se estudian las denominadas funciones lógicas.
Un ejemplo elemental ilustrará de forma rápida y sencilla en qué consisten dichas funciones.

Supongamos que la vaca de las figuras, quiere pasar del pastizal # 1 al pastizal # 2. Para que pueda pasar, tiene que encontrar la puerta A abierta, "y" la B abierta "y" la C también abierta. Si cualquiera de ellas se encuentra cerrada, la vaca no podrá pasar.

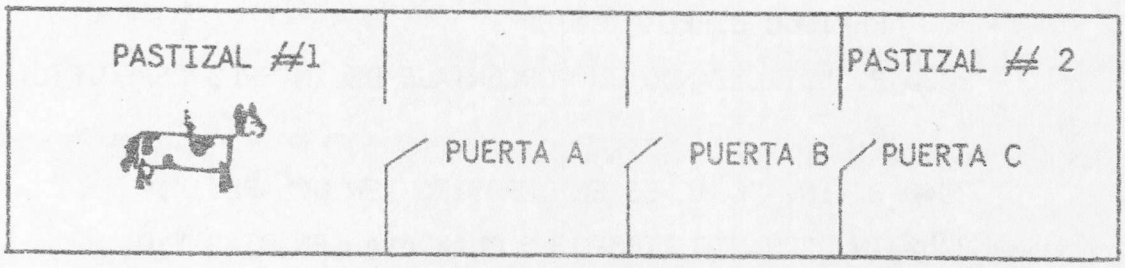


FIGURA 1

A este modo de operar corresponde la llamada "función lógica Y".

En la figura 2, la vaca quiere pasar del pastizal 3 al 4, pero las puertas están colocadas de manera distinta a las anteriores. Para que la vaca pueda pasar, "o" la puerta X está abierta, "o" la Y,"o" la Z, "o" las 3, "o" dos cualesquiera de ellas.

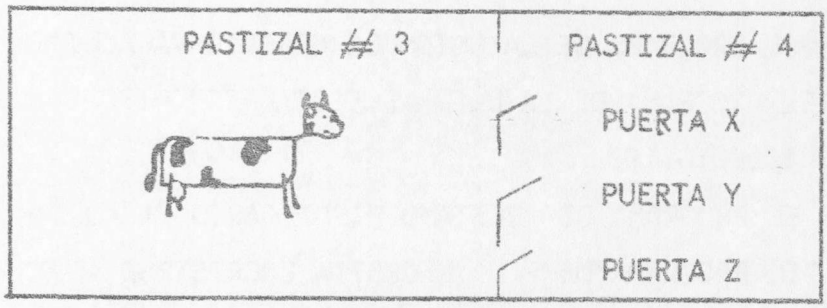


FIGURA 2

Esta es la "función lógica O".

En matemáticas se usa el símbolo + (más) de la suma para la "función lógica o" y el signo X (por de la multiplicación) para la"función lógica y". En nuestro caso entonces tendremos que utilizar el * (asterisco) para la "función lógica y".

A pesar de que al principio el concepto de "símbolos lógicos" puede traer algunas dificultades, su uso es sumamente sencillo. Escribamos :

```
10 S = 1:N = 0
20 IN. "¿ESTA ABIERTA LA PUERTA A?";A
30 IN. "¿ESTA ABIERTA LA PUERTA B?";B
40 IN. "¿ESTA ABIERTA LA PUERTA C?";C
50 PRINT
60 IF (A =1)  *  (B =1)  * (C = 1) THEN 100
70 P. "LA VACA ESTA SITUADA EN EL PASTIZAL # 1"
80 END
100 P. "TODAS LAS PUERTAS ESTAN ABIERTAS. LA VACA PUEDE
    PASAR"
```

... y RUN.

Las respuestas que se deben dar a las preguntas son (S/N) ó (SI/NO). Pueden efectuarse varias ejecuciones para observar cómo trabaja el Y lógico de la línea 60.

Explicación del programa :

La línea 10 inicializa los valores de S y de N en 1 y 0 respectivamente.

Las líneas 20, 30 y 40 dan entrada a las posiciones de las puertas, que pueden estar : abiertas (definidas como igual a 1), o cerradas (definidas como 0). También se podían haber asignado otros valores distintos de 0 y 1 en la línea 10 sometiéndolos a la operación lógica en la línea 60.

La línea 6Ø podríamos decir que es la llave. Podemos leer literalmente, "si la puerta A está abierta, y la puerta B está abierta y la puerta C está abierta, entonces leer la línea 1ØØ; si cualquiera de las puertas está cerrada, leer la línea 7Ø

Un ejemplo en electricidad de lógica "y/o" sería la conexión de una bateria con una lámpara a través de unos interruptores. Si los interruptores están en serie, hemos construido una "función lógica y". Si los interruptores están en paralelo se trata de una "función lógica o".

Si nos fijamos en la línea 6Ø, observaremos que existen paréntesis que encierran el signo =. En general, podíamos haber utilizado símbolos >, < , >=, <=.
Anteriormente hemos usado también estos símbolos para efectuar operaciones, o tomar decisiones pero nunca los habíamos dispuesto entre paréntesis de esta manera. Estos paréntesis con símbolos interiores en un determinado programa, son indicativos de que se está utilizando lógica matemática en el programa.

EJERCICIO 24-1 : Usando el programa anterior como modelo, y el "O" lógico, visto en la figura 2, escribir un programa que determine la situación de la vaca en función del estado de las puertas X, Y y Z.

VARIACIONES LOGICAS

He aquí un programa que utiliza el signo > en lugar del signo = dentro de una prueba lógica. El estudiante pasa si tiene una calificación final superior a 6Ø ó una calificación de medio curso superior a 7Ø y calificación de los trabajos prácticos superior a 75. Escribir el programa y ejecutarlo varias veces para observar cómo trabajan las

pruebas O lógica e Y lógica en la misma línea (4Ø) del programa.

```
1Ø INPUT "CALIFICACION FINAL";F
2Ø INPUT "CALIFICACION DE MEDIO CURSO";M
3Ø INPUT "CALIFICACION DE TRABAJOS PRACTICOS";H
4Ø IF (F>6Ø) + ((M>7Ø) + (H>75)) THEN 7Ø
5Ø PRINT "SUSPENDIDO"
6Ø END
7Ø PRINT "APROBADO"
```

Este programa es una muestra de la utilidad y posibilidades del uso de las funciones lógicas. Los números de las calificaciones reales, por supuesto, pueden ser fijados a cualquier nivel.

El siguiente programa, a título de ejemplo hace uso de signos igual, mayor que y menor que. Determina si los dos números que se han introducido en la computadora son ambos positivos, ambos negativos o tienen signos contrarios.

Analizar el programa, observando los paréntesis. Hay que pensar de forma "lógica" para entender su ejecución. Escribamos y ejecutemos ;

```
1Ø INPUT "EL PRIMER NUMERO ES";X
2Ø INPUT "EL SEGUNDO NUMERO ES";Y
3Ø IF (X>=Ø) * (Y>=Ø) THEN 7Ø
4Ø IF (X<Ø) * (Y<Ø) THEN 9Ø
5Ø PRINT "SIGNOS OPUESTOS"
6Ø END
7Ø PRINT "AMBOS POSITIVOS"
8Ø END
9Ø PRINT "AMBOS NEGATIVOS"
```

Veamos otro ejemplo que utiliza declaraciones gráficas. Intentemos descifrar qué función realiza la línea 4Ø antes de escribir y ejecutar el programa.

```
1Ø  CLS
2Ø FOR X = Ø TO 127
3Ø FOR Y = Ø TO 47
4Ø IF (X>=64) * (Y>=24) THEN 6Ø
5Ø SET(X,Y)
6Ø NEXT Y
7Ø NEXT X
99 GOTO 99
```

¿Qué sucede si se reemplaza el asterisco (*) de la línea 4Ø por un (+)? Después de razonar a qué resultados nos debe conducir la modificación realizada, podemos ejecutarlo para comprobar la veracidad de nuestras conclusiones.

Veamos un ejemplo más de aplicación de utilización de funciones lógicas aplicado al dibujo de líneas. Escribamos y ejecutemos :

```
1Ø CLS
2Ø FOR X = Ø TO 127
3Ø FOR Y = Ø TO 47
4Ø IF (X = 64) + (Y = 24) THEN 6Ø
5Ø SET (X,Y)
6Ø NEXT Y
7Ø NEXT X
99 GOTO 99
```

En este ejercicio se puede adivinar lo que sucedería si se cambiase el + (o) por un *(y) en la línea 4Ø. Después de pensar el resultado ejecutemos el programa con el cambio a que hemos hecho referencia.

Usando la función INT se puede crear un tablero a cuadros. El razonamiento es el siguiente :

En la dimensión horizontal, INT (X/16) * 16 - X es igual a Ø cuando X sea igual a Ø, 16, 32, 48, 64, 8Ø, 96 y 112.

En la dimensión vertical, INT (Y/8) * 8 - Y será cero cuando Y sea igual a Ø,8,16,24,32,40 y 48.

Reemplacemos la línea 4Ø por :

```
4Ø IF ((INT(X/16) * 16-X)=Ø) + ((INT(Y/6) * 6-Y)=Ø) THEN 6Ø
```

... y RUN. El resultado es un tablero con cuadros de 8 por 8.

EJERCICIO 24-2 : Escribir de nuevo la línea 4Ø del programa que dibuja el tablero a cuadros para producir un tablero a cuadros negro sobre blanco en lugar de blanco sobre negro.

Material Aprendido en el Capítulo 24

VARIOS

* usado como símbolo lógico Y.

\+ usado como símbolo lógico O.

SOLUCION A LOS EJERCICIOS PROPUESTOS EN EL CAPITULO 24

RESPUESTA TIPO AL EJERCICIO 24-1 :

Sólo se requieren cambios en dos líneas :

```
6Ø IF (A=1) + (B=1) + (C=1) THEN 1ØØ
1ØØ P. "ESTA ABIERTA ALGUNA PUERTA. LA VACA PUEDE PASAR"
```

La línea 6Ø dice "si la puerta A está abierta, O la puerta B está abierta, O la puerta C está abierta, ir a 1ØØ".

RESPUESTA TIPO AL EJERCICIO 24-2 :

```
4Ø IF ((INT(X/16)* 16-X)<>Ø)* ((INT(Y/8) *8-Y)<>Ø) THEN 6Ø
2Ø FOR X = 1 TO 126
3Ø FOR Y = 1 TO 46
```

Se ha realizado el cambio de las líneas 2Ø y 3Ø para obtener una imagen diferente.

CAPITULO 25

SUBRUTINAS AVANZADAS

En el capítulo 15 se tocó el tema de las subrutinas. Incluso "llamamos" a una para practicar con ellas. Luego volvimos a escribir una subrutina como parte del programa principal y los resultados fueron los mismos. Después de estas consideraciones podríamos preguntarnos : ¿qué tienen entonces de especial las subrutinas? El capítulo 25 tratará fundamentalmente de dar respuesta a esta pregunta.

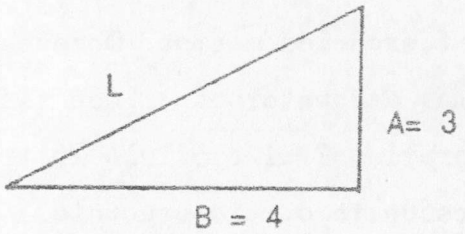
Para refrescar la memoria : Una subrutina es una parte de un programa que la computadora ignora hasta que una declaración GOSUB la manda llamar. Después de ejecutar la subrutina, la computadora devuelve el control del programa al punto inmediato-siguiente de la declaración COSUB (debido al RETUN de la última línea de la subrutina). De esta forma puede llamarse tantas veces como se quiera a una subrutina desde el programa principal, ya que el control del programa regresa siempre al punto de donde partió.

Un ejemplo de utilización de subrutina es el caso de extracción de una raíz cuadrada en un determinado proceso de cálculo. Muchas formas de BASIC (como el BASIC NIVEL II), permiten calcular $\sqrt{X}$ usando simplemente la declaración Y=SQR(X). El nivel 1 no posee esta cualidad y por ello es necesario agregar un programa relativamente simple para lograr la misma cosa, o sea, una subrutina.

La mayoría de los programas de procesos de cálculo requieren el uso de funciones trigonométricas o logarítmicas; por lo tanto se necesitará echar mano de subrutinas, en el caso de que estas funciones no estén contenidas directamente en el BASIC que se esté utilizando.

La mejor forma de entender todo lo que se ha estado diciendo es mediante un ejemplo.

Supongamos que se quiere hallar la hipotenusa de un triángulo rectángulo, conociendo los dos catetos (A=3,B=4). El teorema de Pitágoras dice : $L = \sqrt{A^2 + B^2}$. Se pretende escribir un programa que calcule la hipotenusa para dos valores determinados de los catetos.



Si tuviéramos implementada la función "SQR", correspondiente a la raíz cuadrada, cosa que no sucede en el BASIC I, el programa podría ser :

```
10 REM * SOLUCION A LA RAIZ CUADRADA CON SUBRUTINA *
20 IN. "LA LONGITUD DEL LADO A=";A
30 IN. "LA LONGITUD DEL LADO B=";B
40 L = SQR(A *A+B * B)
50 P. "A","B","L"
60 P. A,B,L
```

Si se escribe y ejecuta el programa, el ordenador responde con :

```
WHAT?
40 L = S?QR (A *A + B * B)
```

Eso indica que el ordenador no reconoce SQR por lo que se deberá llamar a la subrutina raíz cuadrada del apartado 6º (LIBRERIA DE FUNCIONES) para que el programa sea ejecutable.

Vayamos al apartado 6º y localicemos la subrutina de la raíz cuadrada. Existen tres cosas que se deben tener en cuenta cuando se utiliza una llamada a subrutina :

1. ¿Cuál es la variable de entrada?
2. ¿Cuál es la variable de salida?
3. ¿Qué otras variables se utilizan por la subrutina para cálculos internos? Debe ponerse especial cuidado en no usar estas variables internas a la subrutina, en el programa principal.

Una vez escrita la subrutina raíz cuadrada correctamente, efectuemos los siguientes cambios en el programa residente :

```
4Ø X = A* A + B * B:GOSUB 3ØØ3Ø
45 L = Y
```

... y RUN. La computadora, después de una pausa, imprime la respuesta :

A	B	L
3	4	5

En el supuesto de que la ejecución no haya sido la indicada, revisemos el programa principal y la subrutina.

Observaciones :

. La línea 3ØØØØ, es un bloque de protección que evita que la computadora ejecute la subrutina después de completar el programa principal. Siempre se debe colocar esta línea cuando se utilicen subrutinas.

. El motivo por el cual se envía el control a la línea 3ØØ3Ø se debe a que las líneas anteriores (3ØØ1Ø y 3ØØ2Ø), son partes inoperantes del programa; sólo son etiquetas de identificación.

. La línea 4Ø tiene dos declaraciones diferentes. X= A * A + + B *B efectúa el cálculo previo a la extracción de la raíz cuadrada asignándole la letra X, en concordancia con la subrutina. El GOSUB 3ØØ3Ø calcula la raíz.

. La línea 45 asigna a la hipotenusa L el valor de la salida de la subrutina Y. Esta línea se podía haber eliminado utilizando la Y como nombre para la hipotenusa, esto significaría cambiar las declaraciones PRINT de las líneas 5Ø y 6Ø, sustituyendo Y por L.

Antes de continuar, puede resultar interesante guardar la subrutina de la raíz cuadrada en un casette para evitar tener que escribirla de nuevo. Cada programador, puede decidir la conveniencia de guardar todas las subrutinas de forma separada, todas juntas, ó en distintas combinaciones, lo cual permitirá cargar sólo aquellas que se necesiten para un propósito dado.

Aprovechando que tenemos cargada la subrutina "raíz cuadrada", vamos a mostrar un ejemplo práctico para dibujar una serie de circunferencias u óvalos concéntricos. Escribamos y ejecutemos :

```
1Ø CLS
2Ø FOR R = 2 TO 22 STEP 4
3Ø FOR A = -R TO R
4Ø X =R* R-A *A:GOSUB 3ØØ3Ø:Y=INT(Y-.5)
5Ø SET(A+6Ø,23-Y)
6Ø SET(A+6Ø,23+Y)
7Ø NEXT A
8Ø NEXT R
9Ø GOTO 9Ø
```

Explicación :

Como se sabe, la ecuación de la circunferencia en el plano con los ejes centrados es $x^2 + y^2 = R^2$, de donde $y=\sqrt{R^2 - x^2}$. Las abcisas están representadas en el programa por la letra A que varía en cada circunferencia del valor -R al R. Las ordenadas están representadas por la letra Y y su valor se redondea con la función INT. A la suma de cuadrados $R^2 - A^2$ se le asigna la letra X (no confundirla con las abcisas).

La línea 1Ø nos proporciona una pantalla limpia para empezar a trabajar.

La línea 2Ø crea un bucle que incrementa el radio R de 2 a 22 en incrementos de 4.

La línea 3Ø crea un bucle "anidado" que incrementa la abcisa de nuestra gráfica de -R a R.

La línea 4Ø calcula el valor X, extrae la raíz cuadrada y redondea la ordenada a valores enteros.

Las líneas 5Ø y 6Ø centran al círculo sobre la pantalla y lo "dibujan". La línea 5Ø da origen a la parte inferior del círculo, y la 6Ø a la parte inferior.

Las líneas 7Ø, 8Ø y 9Ø nos son ya familiares.

NOTA :

En realidad las circunferencias son óvalos debido a que los puntos son rectángulos en lugar de cuadrados. Para obtener un programa que dibujara circunferencias habría que efectuar alguna modificación. Se recomienda intentar buscar las soluciones por uno mismo.

SALTO ENTRE SUBRUTINAS

Hasta aquí, hemos hecho uso de la declaración GOSUB en, el programa principal para llamar a una subrutina, pero también se pueden hacer llamadas, de una subrutina a otra.

Supongamos que deseamos calcular 3^{11}. Esto es 3 multiplicado por sí mismo 11 veces. Podría hacerse el cálculo multiplicando sucesivas veces 3 *3 *3*3 *3* 3* 3 * 3 * 3 *3 *3* 3, pero, ¿y si lo que se quisiera calcular fuera $3^{11.4}$?, o lo que es lo mismo ¿ 3 elevado a la potencia 11.4?. La simple multiplicación no podría resolver este problema. La solución a esta cuestión se encuentra en la subrutina exponencial.

La subrutina exponencial, en realidad, llama a otras dos subrutinas; una para los logaritmos y otra para las exponenciales. El método deriva de las siguientes relaciones matemáticas :

$$X^Y = e^{\log_n X^Y} = e^{Y * \log_n X}$$

Con la subrutina escrita, todo lo que hemos de hacer es suministrarle los valores de X e Y, en este caso 3 y 11.4, y ella se encarga de obtener la respuesta final.

En el apartado 6º puede encontrarse la subrutina de Exponenciación. Escribirla cuidadosamente añadiendo el programa principal :

```
10 PRINT "OBTENCION DE X ELEVADO A LA POTENCIA Y"
20 INPUT "X =  ";X
30 INPUT "Y =  ";Y
40 GOSUB 30120
50 PRINT "LA RESPUESTA ES";P
60 GOTO 10
```

... y RUN.

(Usar la tecla BREAK en caso de querer salir del programa).

Explicación :

Las líneas 2Ø y 3Ø porporcionan valores para X e Y.

La línea 4Ø transfiere el control del programa principal a la subrutina de exponenciación.

La línea 3Ø14Ø llama a la subrutina LOG para obtener el logaritmo de X.

La línea 3Ø28Ø devuelve L= LOG(X) a la subrutina de exponenciación. (Obsérvese que el control se pasa a la declaración inmediatamente posterior a la orden GOSUB aún cuando la declaración se encuentre sobre la misma línea como en este caso).

La línea 3Ø14Ø llama a la subrutina de expanenciación para calcular e^{Y*L}.

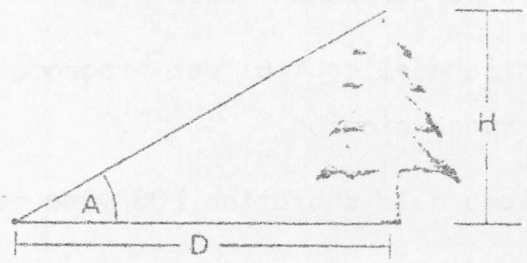
Finalmente, la línea 3Ø15Ø ajusta la magnitud de P y devuelve el valor calculado de $P = X^Y$ al programa principal. La línea 5Ø manda imprimir el valor obtenido.

Observar que la variable X de entrada, cambia de valor en la subrutina; por tanto si se necesitara su valor más adelante, no se podría hacer referencia a dicho valor, porque ya no es el mismo. La manera de resolver este problema sería asignar a otra variable el mismo valor, por ejemplo añadiendo la línea 35 :

```
35 S = X
```

Al agregar esta línea hay que tener en cuenta que el valor de X queda ahora almacenado en la variable S, aue guarda dicho valor para uso posterior. Por todo lo dicho, conviene tener cuidado con las subrutinas que cambian el valor de la variable de entrada.

Otro ejemplo : Se desea encontrar un árbol de Navidad de 28 pies de alto. Esta tarea puede facilitarse haciendo uso de la trigonometría. En efecto, puede calcularse la altura H del árbol de la figura, conociendo la distancia D y el ángulo A.



La relación que puede establecerse es H = D * TAN(A)

TAN(X) es una de las subrutinas disponibles(ver APARTADO 6º). Esta subrutina es una de las más largas debido a que contiene a otras dos, el seno y el coseno.

Después de teclear NEW, escribamos la subrutina tangente (pasos 3Ø3ØØ a 3Ø455). Asegurémonos de agregar un bloque protector END: 3ØØØØ END.

Escribamos a continuación el programa principal :

```
1Ø IN. "A QUE DISTANCIA SE ENCUENTRA LA BASE DEL ARBOL?";D
2Ø IN. "QUE ANGULO EXISTE ENTRE LA PUNTA Y LA BASE DEL
   ARBOL?";A
3Ø X = A:GOSUB 3Ø32Ø
4Ø H = INT(D * Y + .5)
5Ø IF H = 28 THEN 8Ø
6Ø P. "HAY QUE BUSCAR OTRO ARBOL.ESTE TIENE";H; "PIES DE ALTO"
7Ø GOTO 1Ø
8Ø P. "SE PUEDE CORTAR EL ARBOL"
```

Ejecutar el programa usando distintos valores antes de probar con D = 16 y A = 6Ø

Notas sobre el funcionamiento del programa :

La línea 3Ø da a X el valor del ángulo A. Esto se debe a que la subrutina necesita una entrada X. En esta misma línea se manda el control a la subrutina, que calcula el valor de la tangente con variable de salida Y.

La línea 4Ø calcula la altura H que es igual a D por Y (D veces la tangente del ángulo), y luego, redondea la respuesta al entero más cercano.

La línea 5Ø, averigua si hemos encontrado nuestro árbol; si lo hemos hecho, el control del programa pasa a la línea 8Ø, donde se imprime un mensaje adecuado. Si no es así, la línea 6Ø dice que se busque un nuevo árbol, y la línea 7Ø vuelve a empezar el programa.

También existen más formas de resolver el problema. Si se elige una distancia fija a la base, por ejemplo 12 pies, se puede enfocar el problema tratando de encontrar el ángulo necesario para que la altura del árbol sea de 28 pies.

De la fórmula anterior se deduce

H = D* TAN(A); TAN(A) = H/D

De lo que se trata en definitiva es de hallar el arco-tangente de A, que también se encuentra en la librería de funciones (APARTADO 6º).

EJERCICIO 25-1 : Escribir un programa que acepte como entradas la altura del árbol y su distancia al mismo y que calcule el ángulo correspondiente. Usar la subrutina arcotangente del APARTADO 6º.

EJERCICIO 25-2 : Escribir un programa que realice la gráfica del SENO(X), donde X toma valores de 0 a 360º en incrementos de un grado.

Referirse al programa de óvalos concéntricos para darse idea de cómo poner los puntos sobre la pantalla. Recordar que el margen de X y de SENO(X) deberán ajustarse para que quepan en la pantalla de 128 por 48 posiciones.

Material Aprendido en el Capítulo 25

VARIOS

Subrutinas.

"Variables simuladas"

SUBRUTINAS DISPONIBLES EN EL APARTADO 6º :

Raíz cuadrada.
Exponenciación.
Logaritmos (Natural y Común).
Exponencial (Exponentes de e).
Tangente.
Coseno.
Seno.
Arco coseno.
Arco seno.
Arco tangente.
Signo.

SOLUCION A LOS EJERCICIOS PROPUESTOS EN EL CAPITULO 25

RESPUESTA TIPO PARA EL EJERCICIO 25-1 :

```
1Ø INPUT "DISTANCIA DESDE EL ARBOL";D
2Ø INPUT "ALTURA DEL ARBOL QUE SE BUSCA";H
3Ø X = H/D:GOSUB 3Ø66Ø
4Ø PRINT "EL ANGULO REQUERIDO ES";C; "GRADOS".
```

RESPUESTA TIPO PARA EL EJERCICIO 25-2 :

```
1 CLS
1Ø FOR A = Ø TO 36Ø
2Ø X = A:GOSUB 3Ø37Ø
3Ø Y = - Y * 2Ø
4Ø SET (A/3, Y + 22)
5Ø NEXT A
6Ø GOTO 1
```

CAPITULO 26

PROGRAMAS CORRECTORES

Hasta ahora, se conocen tres mensajes de error (WHAT?, HOW? y SORRY), que informan de que algo anda mal. Pero no siempre es fácil detectar dónde se ha cometido el error exactamente. Se evitan errores actuando de forma sistemática pero incluso los mejores programadores los cometen. Sin embargo, existen formas para localizarlos con rapidez.

El primer paso del proceso de "corrección" es aislar el error del programa, que puede ser :

1) Un problema del Hardware.
2) Un problema del Software.

ERRORES DEL HARDWARE

Comenzando con la posibilidad más remota - ¿está trabajando la computadora de forma adecuada?. Lo más probable es que la computadora esté trabajando bien. Existen varias formas de averiguarlo.

Escribamos :

```
PRINT MEM
```

Si no existe programa alguno cargado dentro de la memoria, la respuesta debe ser,

3583

Si existe un programa cargado, la respuesta será algún valor menor que 3583.

Si la respuesta es mayor que 3583 (suponiendo, claro está, que no se tiene agregada más RAM), o si la respuesta es un número negativo, puede haber problemas de hardware.

Entre las soluciones posibles para cualquiera de los casos anteriores está la siguiente : Apagar la computadora, esperar un minuto y volverla a encender. Una vez encendida, teclear NEW y repetir la prueba P.M. anterior. Si los resultados que se obtienen son los mismos, existe un fallo en la memoria RAM ó ROM, lo cual requiere la actuación del servicio de mantenimiento y reparación autorizados.

Una comprobación muy útil puede ser la efectuada por el programa siguiente :

```
1Ø F. X = 1 TO 876:A(X) = X:N.X:F.Y = 1 TO 876:P.A(Y):
2Ø IF A(Y) - A(Y -1) < > 1P. "MALO"
3Ø IN. Y
```

Escribamos el programa haciendo A(Ø) = Ø y ejecutémoslo.

Después de cerca de diez minutos de espera para generar el procedimiento el monitor deberá mostrar

1 2 3 4 5 6 7 8 9 (etc., hasta 876)

Si aparece la palabra "MALO" en la pantalla, se ha encontrado el problema.
Puede resultar interesante guardar este programa de prueba en la cinta casette.

Dentro de los errores del hardware, está el posible fallo de la pantalla de video. La pantalla de video es similar a un televisor. Tiene controles para el brillo, contraste, sincronización horizontal y vertical, etc... En caso de desajuste, existen unos controles internos a la computadora, pero es necesario conocerlos muy bien porque un error podría deteriorar y dejar fuera de servicio los circuitos integrados.

A veces existen problemas en la grabación con casette. Cuando se presenta un caso de este tipo, conviene repasar :

1. ¿Está todo conectado? ¿Correctamente? ¿Firmemente?.
2. ¿Están bien las baterías de la grabadora? (si las tiene)
3. ¿Se ha evitado la retroalimentación a tierra de la grabadora, discutida en el capítulo de grabación de casettes?
4. ¿Se ha fijado a un nivel adecuado el control de volumen?.
5. ¿Está el control del tono de la grabadora en "agudos"?.
6. ¿Se están utilizando instrucciones "legales"?.

Si no se ha encontrado el problema en el hardware, entonces lo más probable es que se encuentre en el software; para salir de dudas, se carga un programa que se sabe que funciona y se ejecuta; si se obtienen resultados positivos, la prueba es definitiva.

ERRORES DEL SOFTWARE

Cuando la computadora da un mensaje de WHAT? o de HOW?, indica normalmente la línea errónea del programa, como en :

```
WHAT?
1Ø X = S?QR(1 - X  X)
```

En el caso de un mensaje WHAT? seguido por una línea del programa, la computadora inserta un signo de interrogación inmediatamente antes del error. En este caso, la computadora no reconoce SQR (recordar que no se tiene la función raíz cuadrada interconstruida en el BASIC NIVEL I) así es que lee 1Ø X = S y luego busca un operador matemático, un (:), un fin de línea o alguna otra continuación válida. "Q" simplemente no se ajusta a lo anterior, por lo tanto la computadora lo trata como si fuera un error.

Con los mensajes HOW?, la computadora inserta un signo de interrogación inmediatamente después del error. Por ejemplo :

HOW?

1Ø PRINT INT(I)?/23

El signo de interrogación nos dice que el error fue descubierto durante la ejecución de la función INT. Podemos adivinar que el valor de I probablemente excedía del rango permisible para la función INT (el valor deberá ser mayor que (-) 32768 y menor a (+) 32768).

Ahora veamos algunas de las fuentes más comunes de "errores detectados por la computadora".

1. Supongamos que el error está en la declaración PRINT ó INPUT.

   a) ¿Están escritas las comillas de comienzo y fin en un texto que las requiere?

      Ejemplo :

      1Ø PRINT "RESPUESTA ES, X:GOTO 5

      ERROR : faltan comillas finales después de ES.

   b) ¿Se ha utilizado un nombre de variable diferente a una sola letra del alfabeto?

      Ejemplo :

      1Ø INPUT AG,S1

      ERROR : Los nombres de las variables del NIVEL*I no podrán tener más de una letra ni una combinación letra/número.

   c) ¿Se han olvidado puntos y comas o comas que separaban variables o un texto, o se ha encerrado al punto y coma o a la coma dentro de las comillas?

      Ejemplo :

      1Ø PRINT "EL VALOR ES;"V

ERROR : El punto y coma está dentro de las comillas (por lo que la "cadena" de palabras y la variable no están separadas adecuadamente).

d) ¿Se ha olvidado el número de línea, se ha intercalado alguna letra con el número o se ha utilizado un número de línea mayor que 32767?

Ejemplo :

```
72B3 PRINT "NUMERO DE LINEA ERRONEO".
  ↑
  └── ERROR
```

e) ¿Se han escrito accidentalmente dobles comillas en el texto?

Ejemplo :

```
1Ø PRINT ""DIJO HOLA"
```

f) ¿Se ha escrito alguna línea de más de 7Ø caracteres de larga?

g) ¿Se ha escrito mal PRINT o INPUT?

h) ¿Se ha introducido involuntariamente algún carácter indeseable en la línea, especialmente alguna coma extra o un punto y coma de más?.

2. Si el error está en una declaración READ, puede ser debido a cualquiera de las razones apuntadas o a cualquiera de las siguientes :

a) ¿Existe realmente una declaración DATA para que lea la computadora? Recordar, que sólo se leerá una información una vez, a menos que sea restaurada por un RESTORE.

Ejemplo :

```
1Ø READ X,Y,Z
2Ø DATA 2,5,
```

ERROR : Sólo existen dos números para ser leídos por la computadora. Si se quiere que Z valga cero, debe especificarse.

```
2Ø DATA 2,5,Ø
```

3. Si la zona de error corresponde a un bucle FOR-NEXT, la mayoría de las posibilidades anteriores también son aplicables, además de :

b) ¿Existe alguna declaración NEXT para completar la FOR?

c) ¿Están presentes todos los requisitos del bucle, a saber : un comienzo, un final, un nombre de variables, y el tamaño del incremento si es que no es igual a 1?

Ejemplo :

```
1Ø A = 1 TO N
```

ERROR : Son necesarios un FOR y UN NEXT.

d) ¿Se han anidado accidentalmente dos bucles usando la misma variable en ambas?

Ejemplo :

```
1Ø FOR X = 1 TO 5
2Ø FOR X = TO 3
3Ø PRINT X
4Ø NEXT X
5Ø NEXT X
```

ERROR : Los bucles anidados deberán tener diferentes nombres, de variables.

e) ¿Tiene la variable interior del bucle la misma letra que el contador del mismo?

Ejemplo :

```
1Ø  A = 22
2Ø  FOR R = 1 TO 5
3Ø   R = 18
4Ø     Y = R*A
5Ø       PRINT Y
6Ø  NEXT R
```

ERROR : El valor de R se ha cambiado dentro del bucle por otro valor distinto y NEXT R se ha sobrepasado, ya que 18 es mayor que 5.

f) ¿Existen bucles anidados incorrectamente, de modo que un bucle no se halle dentro de otro por completo, sino tan sólo de forma parcial?

Ejemplo :

```
1Ø FOR X = 1 TO 6
2Ø  FOR Y = 1 TO 8
3Ø   SET(X,Y)
4Ø  NEXT X
5Ø NEXT Y
```

4. Si la declaración errónea está dentro de un IF-THEN o GOTO

a) ¿Existe realmente el número de línea especificado por el THEN o EL GOTO? Hay que cuidar este error especialmente cuando se elimina una línea en el proceso de "mejorar" ó "limpiar" un programa.

5. El error nos viene como un SORRY pero el P.M. nos indica que hay espacio en la memoria : Si se obtiene un SORRY y se está utilizando un vector A(X), asegurarse de verificar el P.M. luego restar 4 bytes por cada elemento del vector utilizado. Probablemente se sobrepasa la memoria disponible.

6. El error se manifiesta como un HOW? y la línea del programa que contiene el error es impresa con un signo de interroga-

ción que aparecerá en algún lugar de la misma.

a) ¿Se sobrepasan los límites de alguna de las funciones construidas?

b) ¿Sobrepasa alguno de los valores de la línea el tamaño máximo o mínimo para números del NIVEL I?

c) ¿Se ha pretendido que la computadora dividiese por cero?

Para averiguar si es que se hizo cualquiera de las cosas anteriores, imprimir los valores de todas las variables que se usaron dentro de la línea errónea. Si aún no se ve el error, intentar efectuar las operaciones que se indican en la línea. Por ejemplo, el error puede ocurrir durante la multiplicación de dos números muy grandes.

Por supuesto que estos no son todos los errores posibles que pueden aparecer, pero por lo menos dan alguna idea de cómo hallarlos. Ya que no podemos evitar completamente los errores tontos, es muy conveniente eliminarlos cuanto antes.

En definitiva, todo lo que la computadora puede decir es si hemos o no seguido todas sus reglas. Pero además, podemos haberlas seguido a la perfección y estar cometiendo errores.

Entre los errores no detectados por la computadora se encuentran :

1. El olvidar inicializar variables (iniciándose éstas con los valores antiguos). Recordar que no se puede suponer que las variables no utilizadas valen cero.

2. El reinicializar una variable accidentalmente, (particularmente fácil cuando se están usando bucles).

Ejemplo :

```
1Ø FOR N = 1 TO 3
2Ø  READ A
3Ø  PRINT A
4Ø  RESTORE
5Ø NEXT N
6Ø DATA 1,2,3
```

3. El invertir condiciones, esto es, el usar "=" cuando se quiere "<>", o "mayor que" cuando se quiere decir "menor que".

4. El incluir "iguales", como en "menor o igual a", cuando en realidad lo que quería decir era sólo "menor que".

5. El confundir variables de nombres similares, principalmente la variable A, la cadena A$, y el vector A(X). No están relacionados.

6. Olvidar el orden de la ejecución del programa de izquierda a derecha en cada línea, (las multiplicaciones y las divisiones siempre tienen prioridad sobre la adición y la sustracción y las funciones intrínsecas (INT,RND,ABS,etc) tienen prioridad sobre todo lo demás).

7. El contar incorrectamente en los bucles FOR I = Ø TO 7 hace que la iteración se ejecute ocho veces y no siete.

8. Usar las mismas variables accidentalmente en dos lugares diferentes. Esto es correcto siempre y cuando no se necesite la variable antigua posteriormente, pero desastroso si se necesita después.
   Se debe tener especial cuidado cuando se combinan programas o al usar las subrutinas del apartado 6º.

¿Pero cómo se detectan estos errores, si la computadora no los imprime?. Las reglas a seguir son :

1. AISLAR el error. Insertar "banderas" temporales. Agregar declaraciones extras STOP, END y PRINT hasta conseguir que el error se halle contenido en una o dos líneas. Ejemplos de banderas útiles :

```
299 PRINT "                                    LINEA # 299"
399 IF X < Ø THEN PRINT "X FUERA DE MARGEN EN # 399":
    STOP
```

   La línea 299 ayuda a seguir la trayectoria del programa. La línea 399 se usa para localizar el punto donde X se sale de rango.

2. Efectuar "pruebas" lo más simple posibles. No agregar complicaciones hasta que se haya encontrado el error.

3. Verificar los casos simples a mano para comprobar su lógica, dejando que la computadora realice el trabajo difícil. No tratar de resolver problemas complejos en papel, porque fácilmente introduciremos más errores que los que queremos evitar. Usar la modalidad de calculadora, u otra calculadora de bolsillo para trabajar.

4. Recordar que se puede forzar a la computadora para que comience en cualquier línea del programa que se desee; para ello se teclea RUN # # # (donde # # # representa a la línea deseada). De esta forma se puede ir repasando el programa; se dan valores a las variables usando declaraciones en el modo de calculadora y se ejecuta el programa desde el punto de comienzo escogido. También se puede intercalar un STOP, en un punto posterior. Así, se pueden probar partes del programa, separadamente del resto.

5. Recuérdese que también se puede listar una parte del programa sin necesidad de listarlo completo (LIST # # #).

6. Practicar la "programación defensiva". Que el programa funcione "bien" no implica que sea de confianza. Los programas que aceptan datos de entrada y que los procesan, pueden ser particularmente engañosos. Es preciso acostumbrarse a verificar un programa nuevo en todos sun puntos críticos. Ejemplos : Un programa de raíz cuadrada deberá ser revisado para ver si contiene entradas menores o iguales a cero. Las funciones matemáticas que se hayan programado, deberán ser verificadas en aquellos puntos donde la función no esté definida, tal como TAN(90º).

Finalmente hay que reseñar la importancia del diagrama de flujo como guía que disminuye la probabilidad de cometer errores y facilita su detección.

Material Aprendido en el Capítulo 26

VARIOS

Programación defensiva.

Errores detectados por la computadora.

Banderas.

Procedimientos para la revisión del hardware.

RESUMEN DEL NIVEL BASIC I

Ordenes	Propósito	Ejemplo	Capit.
NEW	Borra todas las líneas de programa almacenado en memoria.	NEW(no es parte del programa)	1
RUN	Inicia la ejecución del programa en el número de línea más bajo.	RUN (no es parte del programa) RUN (no es parte del programa).	1
RUN # # #	Inicia la ejecución del programa en el número de línea especificado.	RUN 3ØØ (no es parte del programa).	11
LIST	Muestra las primeras doce líneas del programa almacenado en la memoria, empezando por el número de línea más bajo. Pulsando una tecla, se detiene el listado.	LIST (no es parte del programa).	2
LIST # # #	Lo mismo que LIST, pero empieza por un número de línea especificado.	LIST 3ØØ (no es parte del programa).	11
CONT	Continúa la ejecución del programa cuando aparezca BREAK AT # # # .	CONT (no es parte del programa).	

Declaraciones	Propósito	Ejemplo	Capit.
PRINT	Imprime el valor de una variable o expresión; también imprime lo que se encuentre dentro de "comillas".	1Ø PRINT "A+B=";A+B	1,2,3
INPUT	Le dice a la computadora que nos permita introducir datos desde el Teclado.	1Ø INPUT A,B,C	7
INPUT	Se puede usar simultáneamente con la declaración PRINT.	1Ø INPUT "ESCRIBIR UN NUMERO";A	7
READ	Lee datos de una declaración.	1Ø READ A,B,C,A$	16

DATA	Guarda datos para ser leídos por una declaración READ.	2Ø DATA 1,2,3"SALLY"	16
RESTORE	Hace que la siguiente declaración READ empiece con el primer elemento en la primera línea DATA.	3Ø RESTORE	16
LET	(Opcional) le asigna nuevos valores a la variable.	1Ø LET A=3.14159	2
GOTO	Transfiere el control del programa a la línea del programa designada.	1Ø GOTO 3ØØ	6
IF-THEN	Establece un punto de prueba.	1Ø IF A=B THEN 3ØØ	6
FOR-NEXT	Establece un bucle para que sea ejecutado un número especificado de veces.	1Ø FOR I=1 TO 1Ø 2Ø NEXT I	10,11,13
STEP	Especifica el tamaño del incremento para ser utilizado por las iteraciones FOR-NEXT.	1Ø FOR I=Ø TO 1Ø STEP 2	1Ø
STOP	DETIENE LA EJECUCION DEL PROGRAMA e imprime un mensaje BREAK AT # # #	1Ø IF A = B STOP	11
END	Finaliza la ejecución del programa y fija el contador del programa en cero.	99 END	2
GOSUB	Transfiere el control del programa a la subrutina que se inicia en la línea especificada.	1Ø GOSUB 3ØØØ	15,25
RETURN	Finaliza la ejecución de la subrutina y devuelve el control a la línea siguiente a GOSUB.	3Ø1Ø RETURN	15,25
ON	Ramificación de caminos múltiples utilizada por las declaraciones GOTO y GOSUB.	1Ø ON N GOTO 3Ø,4Ø,5Ø 1Ø ON N GOTO 3ØØØ,4ØØØ,5ØØØ	15

	Propósito	Ejemplo	Capítulos
AT	(sigue a PRINT) Empieza la impresión en un lugar especificado de la pantalla.	1Ø PRINT AT 65Ø "HOLA"	22

TAB	(sigue a PRINT) Empieza a imprimir después de un número especificado de espacios desde el margen izquierdo.	1Ø PRINT TAB(1Ø): "MES";TAB(2Ø); "RECIBOS"	12

Gráficas	Propósito	Ejemplo	Capit.
SET	Ilumina un lugar especificado de la pantalla.	1Ø SET(3Ø,4Ø)	20,22
RESET	Apaga un lugar especificado de la pantalla.	2Ø RESET (3Ø,4Ø)	20,22
POINT	Verifica el lugar gráfico especificado: si el punto está "encendido" obtiene un 1; si está"apagado" obtiene un Ø.	3Ø IF POINT(3Ø,4Ø) =1 THEN PRINT "ENCENDIDO"	22
CLS	Apaga todas las posiciones gráficas (limpia la pantalla).	1Ø CLS	10,20

Funciones Implementadas	Descripción	Ejemplo	Capit.
MEM	Nos da el número de bytes libres restantes en la memoria.	1Ø PRINT MEM	8
INT(X)	Obtiene el mayor entero que sea menor que o igual a X.	1Ø I = INT(X)	14
ABS(X)	Valor absoluto de X	1Ø M = ABS(A)	17
RND(Ø)	Nos da un número al azar entre Ø y 1.	1Ø X = RND(Ø)	19
RND(N)	Nos da un número al azar entre 1 y N.	1Ø X = RND(5ØØ)	19

Operadores Mate.	Función	Ejemplo	Capit.
+	Adición	A + B	3
-	Sustracción	A - B	3
*	Multiplicación	A * B	3
/	División	A/B	3
=	Le asigna el valor de la parte derecha a la variable de la parte izquierda.	A=B	3

Operadores Relacionales	Relación	Ejemplo	Capt.
<	es menor que	A B	6
>	es mayor que	A B	6
=	es igual a	A = B	6
<=	es menor o igual a	A =B	6
>=	es mayor o igual a	A =B	6
<>	no es igual a	A B	6

Operadores Lógicos	Función	Ejemplo	Capit.
*	AND(Y)	(A=3)*(A=7)"A es igual a 3 y A es igual a 7".	24
+	OR(O)	(A=3)+(A=7)"A es igual a 3 ó A es igual a 7".	24

Variables	Propósito	Ejemplo	Capit.
de la A a la Z	Toma valores numéricos	A = 3.14159	3
A$ y B$	Toma valores encadenados	A$= radio shack	16
A(X)	Almacenar los valores de un vector.	A(Ø) = 4ØØ	21

ABREVIATURAS DEL NIVEL BASIC I

Comando/Declaración	Abreviatura	Comando/Declaración	Abrevit.
PRINT	P.	TAB (después de PRINT)	T.
NEW	N.	INT	I.
RUN	R.	GOSUB	GOS.
LIST	L.	RETURN	RET.
EBD	E.	READ	REA.
THEN	T.	DATA	D.
GOTO	G.	RESTORE	REST.
INOUT	IN.	ABS	A.
MEM	M.	RND	
FOR	F.	SET	S.
NEXT	N.	RESET	R.
STEP (después de FOR)	S.	POINT	P.
STOP	ST.	PRINT AT	P.A.
CONT	C.		

APARTADO 3º

PROGRAMAS DE JUEGOS

CAPITULO 27

ARCHIVO DE DATOS EN CASSETES

Este capítulo, aunque puede ser opcional, es muy importante debido a que supone una innovación sobre lo que se ha visto hasta ahora.
Es lo que podemos llamar el paso de la programación al mundo más amplio del procesamiento de datos.

Hasta ahora, se ha dependido de las 26 variables numéricas del nivel I, de la A a la Z, 876 posiciones para los vectores A(X), dos variables encadenadas A$ y B$, y las líneas DATA para almacenar los datos que necesitan nuestros programas. Esto tiene dos limitaciones

1. La memoria de la computadora puede no ser lo suficientemente grande como para retener toda la información que se necesita (por ejemplo una lista de inventario).
2. Cuando se apaga la computadora los valores de las variables se pierden.

Los archivos de datos en cassete, solucionan ambos problemas. Podemos almacenar tremendas cantidades de información en una cinta y recuperarlas más tarde, de manera parecida a como se guardan los programas.

El trasvase de programas de la microcomputadora al cassete y viceversa se realizaba mediante las instrucciones CSAVE y CLOAD. Para trasvasar datos en lugar de las instrucciones citadas se utilizan

oprimir las teclas RECORD y PLAY de la grabadora simultáneamente; luego teclear y ejecutar el siguiente programa :

```
5Ø  A=1:B=2:C=3
1ØØ PRINT #  A;",";B;",";C
```

Se puede observar la puntuación requerida para separar cada variable para imprimir sobre la cinta, se debe insertar (;",";) entre dos variables de una declaración PRINT #

El proceso de grabación al ejecutar el programa es el siguiente :

1. La grabadora automáticamente empieza (suponiendo que está en la modalidad RECORD).
2. Sobre la cinta se escriben los valores de A,B y C.
3. Cuando se han escrito los valores, la grabadora se detiene; entonces debe oprimirse STOP para desacoplar la cabeza de grabación.

De esta forma se guardan estos valores en cinta.

Para leer de nuevo los datos del cassete, se rebobina la cinta oprimiendo la tecla REWIND hasta el punto donde empezó el archivo de datos. Luego puede editarse el siguiente programa (después de haber borrado el anterior con NEW) :

```
1ØØ  A=Ø:B=Ø:C=Ø
11Ø  INPUT #  A,B,C
12Ø  PRINT "LOS DATOS HAN SIDO LEIDOS DE LA CINTA"
13Ø  PRINT "A=";A,"B=";B,"C=";C
```

Una vez hecho esto, se oprime la tecla PLAY de la grabadora y se ejecuta el programa (RUN).

Si los datos del programa anterior se han leido correctamente, y así mismo fueron correctamente almacenados, la computadora debe mostrar :

LOS DATOS HAN SIDO LEIDOS DE LA CINTA
A= 1
B= 2
C= 3
READY

- La línea 1ØØ fija las variables a cero. De esta forma, si los datos no se leen correctamente, el valor de las variables será cero.

- La línea 11Ø permite que la grabadora comience a funcionar, cargando los tres números en las variables A,B,C. Una vez leídos los 3 números, la grabadora se detiene.

- La línea 12Ø imprime un mensaje que da seguridad. Estos tipos de mensajes son muy utiles como guía para el programador. Por ejemplo, antes de que la computadora ejecute una declaración PRINT #, se puede hacer que imprima un mensaje que avise que hay que poner la grabadora en la modalidad de grabación.

- La línea 13Ø imprime los datos que han sido leídos de la cinta.

NOTA :

Si la grabadora no está en la modalidad de reproducción (PLAY, con las conexiones apropiadas), cuando ejecute una declaración INPUT #, la computadora tratará de leer la cinta hasta que obtenga algo y se perderá el control de la computadora desde el teclado.
La única forma de volver a controlarla es apretando el botón de reajuste (Reset) localizado en la abertura de la parte trasera izquierda del teclado.
Esto termina con el programa pero no lo borra.

Si se imprime con PRINT # una lista de un determinado número de valores sobre la cinta, se debe haber introducido con un INPUT # el mismo número de valores. Si no concuerda el número de elementos de la declaración PRINT # con el número de elementos de la declaración INPUT #, se acabará perdiendo datos o perdiéndose el control de la computadora.

El siguiente programa demuestra como se puede usar un archivo de datos, para crear una lista de elementos de dato, su procesamiento y actualización. Se recomienda su estudio para entender como podrían programas similares manejar inventarios o cualquier otra lista secuencial.

```
  1  REM * TEMP Y HUMEDAD PROM. USANDO UN ARCHIVO DE DATOS. *
  5  C=Ø:CLS
  7  B=Ø
 1Ø  IN. "¿QUE DIA DEL MES ES?";D
 2Ø  IN. "¿CUAL FUE LA TEMPERATURA HOY?";T
 3Ø  IN. "¿CUAL FUE LA HUMEDAD?";H
 4Ø  IF. D=1 THEN 16Ø
 5Ø  P. "CARGUE LAS TEMPERATURAS Y HUMEDADES ANTERIORES DEL MES".
 53  P. "PRIMERO REGRESE CINTA AL PRINCIPIO DE ARCHIVO DE DATOS".
 55  P. "LUEGO OPRIMA TECLA PLAY DE GRABADORA".
 6Ø  IN."OPRIMA ENTER CUANDO ESTE LISTO";A $
 7Ø  FOR X = 1 TO D-1
 8Ø  INPUT #  Y,Z
 9Ø  B = B + Y
1ØØ  C = C + Z
11Ø  NEXT X
12Ø  B = (B + T)/D
13Ø  C = (C + H)/D
14Ø  CLS:P. "LA TEMPERATURA PROMEDIO ES";B
15Ø  P. "LA HUMEDAD PROMEDIO ES";C
16Ø  P.:P. "AHORA LA TRS-8Ø ESCRIBIRA"
17Ø  P. "LA TEMPERATURA Y HUMEDAD DE HOY"
18Ø  P. "SOBRE LA CINTA"
19Ø  P. "ASI QUE OPRIMA LAS TECLAS RECORD Y PLAY"
2ØØ  P. "PERO NO REGRESE LA CINTA"
21Ø  IN. "OPRIMA ENTER CUANDO ESTE LISTO"; A $
22Ø  PRINT #  T;",";H
23Ø  P.:P. "AHORA LA INFORMACION DE HOY SE AGREGO AL ARCHIVO EN
     CINTA"
24Ø  P.:P. "POR FAVOR OPRIMA LA TECLA STOP DE LA GRABADORA"
25Ø  END
```

- La línea 8Ø lee todos los números o valores de los días anteriores, de dos en dos. Cuando se ha leído toda la información, se calculan el promedio de humedad y temperatura (usando la información del día también).
- La línea 22Ø escribe a continuación la información del día en la cinta.

Se puede ejecutar este programa empezando por el primer día del mes.

EJERCICIOS PARA EL USO DE LOS ARCHIVOS DE DATOS

1. ENSEÑANZA, EXAMENES. Escribir un programa que haga un exámen de opción multiple incluyendo diez preguntas. El programa debe escribir los nombres de los estudiantes, y las diez respuestas en un archivo de datos en cassete. Diseñar un programa de manera que pueda tomarse uno cualquiera de los exámenes de los estudiantes. Incluir instrucciones sobre cuándo se deben utilizar las teclas RECORD, PLAY y STOP.

   El programa evaluador debe tomar los datos del archivo creado con anterioridad para leer el nombre y las respuestas del estudiante. Después de un intervalo pasará al siguiente estudiante hasta completar la lista.

2. INVENTARIO. Escribir un programa que cree un arreglo en el cual se almacene la siguiente información sobre un grupo de coches.

   Placas Nº Tamaño del motor Código del color Modelo

   Este programa debe almacenar el arreglo en un archivo de datos, y tiene que realizar las siguientes funciones :

1. Se le puede preguntar en que coche se está interesado, (tecleando el número de placa).
2. Lea el archivo hasta que encuentre el número deseado.
3. Imprima toda la información de ese coche en particular.

NOTA :

Puede consultarse el capítulo 21 donde se desarrolló este mismo arreglo.

Material aprendido en el Capítulo 27

DECLARACIONES	VARIOS
INPUT #	Ficheros de datos
PRINT #	

APARTADO 3º

JUEGOS

INTRODUCCION AL APARTADO

Como complemento al manual de programación BASIC NIVEL I hemos introducido este apartado con objeto de dar unas orientaciones sobre el modo de elaborar programas de cierta complejidad. Se ha seguido el criterio de programar una serie de juegos en relación creciente de dificultad.

Este tipo de problemas es muy adecuado para el aprendizaje de la programación, dada la múltiple variedad de casos y posibilidades que pueden presentarse en ellos.
Así mismo, en este apartado podrá verse la importancia de lo que se llama "programación estructurada".

Esta idea, método de elaboración progresiva de un programa, cobra mayor importancia cuanto mayor es la dificultad del programa. En síntesis se trata de preguntarse que es lo que hay que hacer antes de pensar en como hacerlo y así ir elaborando diagramas de flujo generales que explican el programa en su conjunto.

Los pasos siguientes consistirán en ir subdividiendo cada bloque del diagrama de flujo general en diagramas de flujo más concretos hasta llegar al diagrama final o al programa final. A este proceso progresivo los programadores le denominan "refino".

PROGRAMA DE VELOCIDAD DE REFLEJOS

BASES DEL PROGRAMA

Este sencillo ejercicio trata de elaborar un programa que sea capaz de medir la velocidad de respuesta a una orden visual. Debido a su sencillez no parece necesario un dibujo del diagrama de flujo para su comprensión.

PROGRAMA

```
5 CLS
10 PRINT'EN SUS MARCAS.....'
20 FOR B=1 TO 500
30 NEXT B
40 PRINT
50 PRINT
60 PRINT
70 PRINT TAB(30);'LISTOS.....'
80 X=RND(1500)
90 FOR N=1 TO X
100 NEXT N
110 PRINT
120 PRINT
130 PRINT
140 PRINT
150 PRINT
160 PRINT'.....!YA!.....'
170 CLS
180 FOR Z=10 TO 1 STEP-1
190 PRINT Z
200 NEXT Z
210 PRINT
220 PRINT
230 PRINT
240 PRINT'LE FALTAN REFLEJOS'
250 END
```

EXPLICACION DEL PROGRAMA

- Las líneas de la 1Ø a la 7Ø preparan al jugador.
- Como nota destacable del programa, la línea 8Ø elige al azar el tiempo que tarda en aparecer el YA! (línea 17Ø).
- Las líneas 18Ø a la 2ØØ dan la puntuación del jugador.
- La línea 24Ø indica que se ha sobrepasado el tope estipulado de reflejos.

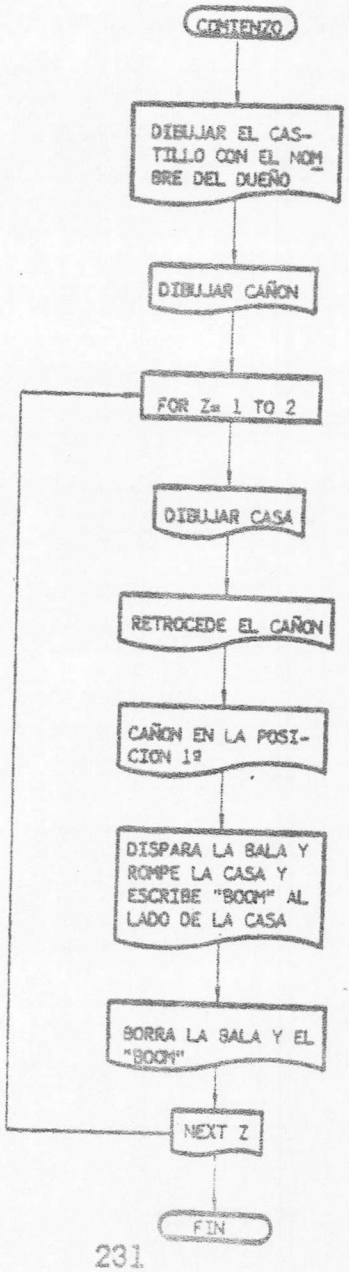
TIRO DEL CASTILLO

BASES DEL PROGRAMA

Este programa es un ejemplo de manejo de las funciones gráficas. El difujo que se quiere conseguir consiste en un castillo que tiene un cañón en la almena. El cañón dispara una bala que rompe una casa. Este ciclo se repite dos veces.

El siguiente diagrama de flujo muestra las fases y desarrollo de creación del programa.

DIAGRAMA DE FLUJO



Además se introducen 3 retardos. Uno de 0,2 segundos después de dibujar la casa, otro de 0,2 seg. después de retroceder el cañón y otro de 2 segundos después de dibujar "BOOM".

A continuación se da el listado correspondiente.

PROGRAMA

```
5 REM * TIRO DEL CASTILLO *
10 INPUT"ESCRIBA SUS INICIALES";A$
20 CLS
30 Z=74
40 FOR Y=17 TO 47
50 FOR X=Z TO 127
60 SET(X,Y)
70 NEXT X
80 IF Y<23 THEN Z=Z+2
200 NEXT Y
210 FOR X=75 TO 123 STEP 4
220 SET(X,16)
230 SET(X+1,16)
240 NEXT X
250 Q=0
300 FOR X=95 TO 125 STEP5
310 FOR Y=47 TO 35 STEP-1
320 RESET(X,Y)
330 NEXT Y
340 NEXT X
400 FOR X=95 TO 125
410 RESET(X,34)
420 NEXT X
500 PRINT@688,"CAS.DE ";A$;
600 FOR X=73 TO 100
610 SET(X,12)
620 SET(X,13)
630 NEXT X
700 FOR X=85 TO 95
710 SET(X,14)
720 SET(X,15)
730 NEXT X
740 RESET(90,13)
750 RESET(91,13)
1000 FOR Z=1 TO 2
1010 FOR X=2 TO 14
1020 FOR Y=40 TO 43
1030 SET(X,Y)
1040 NEXT Y
1050 NEXT X
1100 FOR X=3 TO 13 STEP 2
1110 RESET(X,41)
1120 NEXT X
1130 RESET(7,43)
1140 RESET(8,43)
1190 REM * ESTO HACE RETROCEDER LA BATERIA *
```

```
1200 FOR I=1 TO 100
1210 RESET(73,12)
1220 RESET(73,13)
1230 RESET(74,12)
1240 RESET(74,13)
1250 SET(101,12)
1260 SET(101,13)
1270 SET(102,12)
1280 SET(102,13)
1290 FOR X=1 TO 100
1300 SET(74,12)
1310 SET(74,13)
1320 SET(73,12)
1330 SET(73,13)
1340 RESET(102,12)
1350 RESET(102,13)
1360 RESET(101,12)
1370 RESET(101,13)
1500 FOR X=74 TO 2 STEP-1
1510 P=X-73
1520 Y=P*P/150+12
1530 SET(X,Y)
1540 SET(X-1,Y)
1600 RESET(X+1,Q)
1610 RESET(X,Q)
1620 Q=Y
1630 NEXT X
1640 PRINT@771,"BOOM"
1700 GOSUB 1900
1710 FOR X=1 TO 18
1720 RESET(X,45)
1730 RESET(X,36)
1740 RESET(X,37)
1750 NEXT X
1800 NEXT Z
1810 PRINT@0," ";
1820 END
1900 FOR X=1 TO 1000
1910 NEXT X
1920 RETURN
```

RELACION DIAGRAMA DE FLUJO - PROGRAMA

Dibujo del castillo : líneas de la 4Ø a la 5ØØ

" del cañón : " " " 6ØØ a la 75Ø

" de la casa : " " " 1Ø1Ø a la 114Ø

Retardo 0,2 segundos : línea 12ØØ

Dibujo del retroceso del cañón : líneas de la 121Ø a la 128Ø

Retardo de 0,2 segundos : línea 129Ø

Dibujo del avance del cañón : líneas de la 13ØØ a la 137Ø

" del disparo " " : " " " 15ØØ " " 164Ø

Retardo de 2 segundos : líneas 17ØØ, 19ØØ, 191Ø, 192Ø

Borrado de la bala y el "BOOM" : líneas de la 171Ø a la 175Ø

La repetición del ciclo se realiza merced a las líneas 1ØØØ y 18ØØ.

JUEGO DE DADOS

NORMAS DEL JUEGO

1º) El jugador tira los dados. Si en la primera tirada, la suma da (2,3, ó 12) pierde, si obtiene un 7 o un 11 gana. En cualquiera de estos dos casos, el juego finaliza.

2º) Si la suma de la primera tirada da un número distinto de los anteriormente referidos, éste se convierte en una puntuación que en el caso de que se repita en posteriores jugadas, el jugador gana, mientras que si en alguna de ellas saca un 7, pierde.

Una observación de interés en todos estos programas de juegos, está en la elección de la impresión de resultados en la pantalla de video.

En el caso concreto que aquí se trata, se dibujan dos dados con sus puntuaciones en la parte central de la pantalla. También se imprime la primera tirada, así como una serie de indicaciones y mensajes en la parte superior de la misma.

La explicación del programa viene dada por el diagrama de flujo a través del cual puede seguirse el listado, que no es sino una materialización de aquel en lenguaje BASIC. El bloque "DIBUJAR DADOS" da lugar a su vez a otro diagrama, e incluso cada subrutina de este segundo diagrama puede representarse mediante otro diagrama de flujo. Este caso constituye un típico ejemplo de programación estructurada.

DIAGRAMA DE FLUJO GENERAL

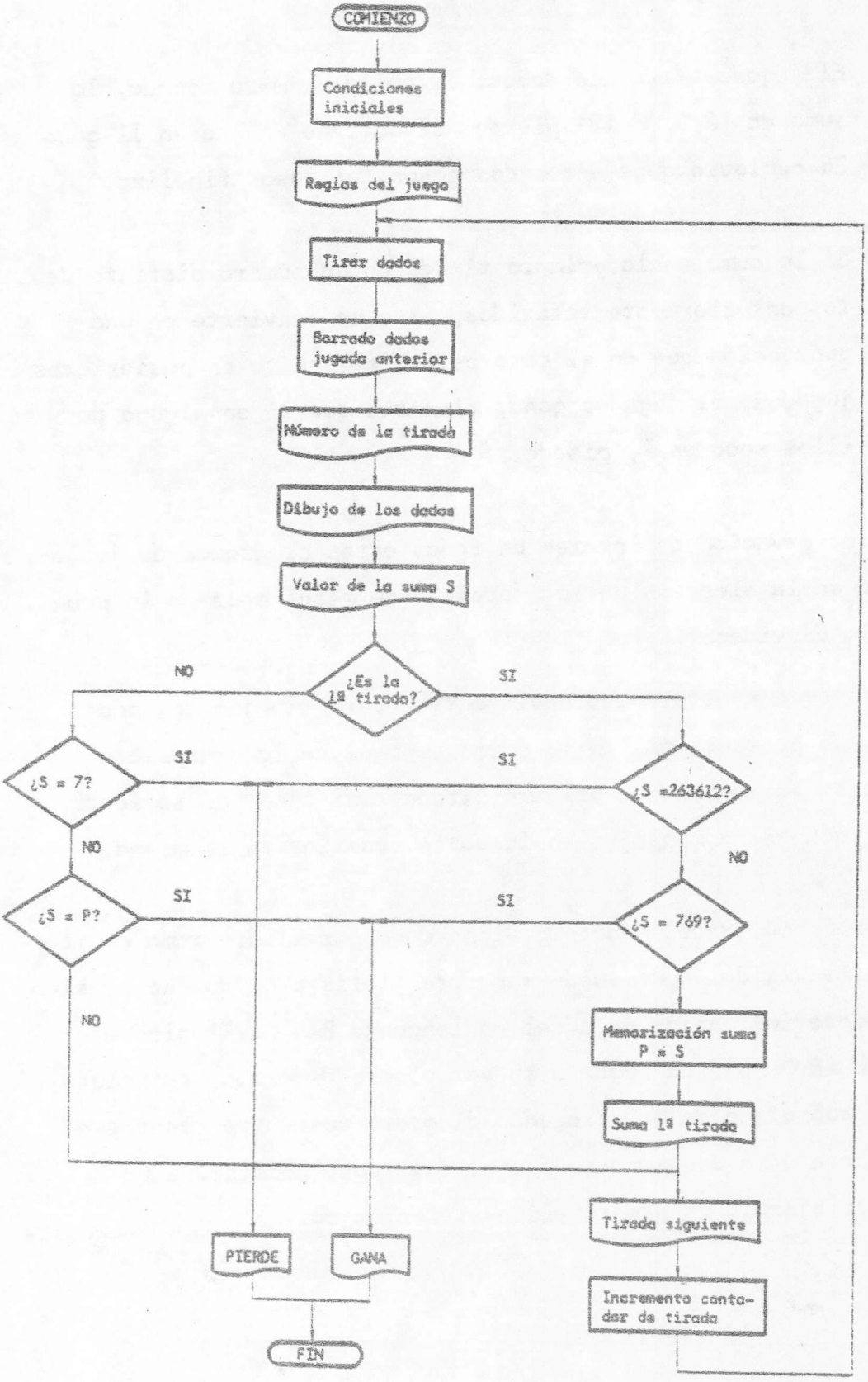
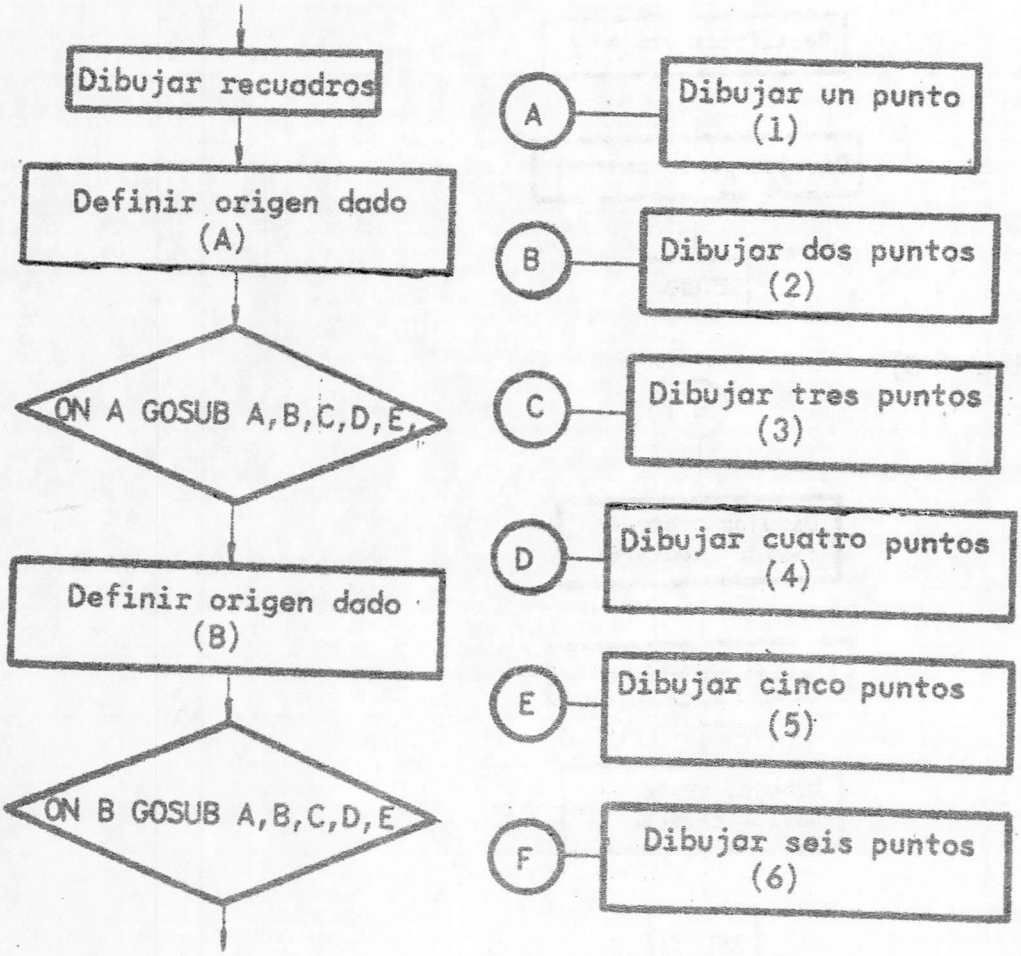


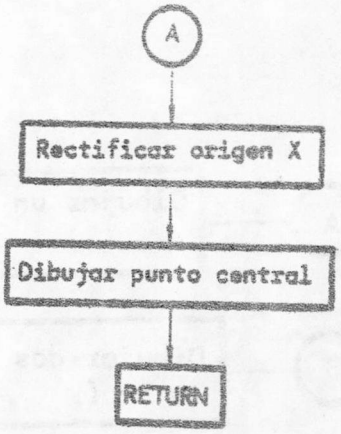
DIAGRAMA DE FLUJO DEL BLOQUE "DIBUJAR DADOS"



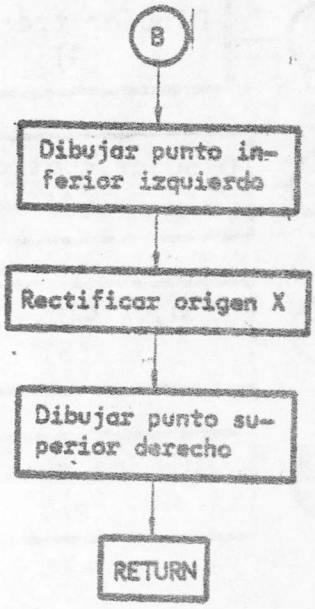
Desde luego existen muchas formas de programar el dibujo de los dados. Se trata entonces de buscar la más sencilla y con el máximo ahorro de instrucciones. Los diagramas de flujo de las subrutinas muestran una manera de llevar a buen término este objetivo pero, por supuesto no es la única.

DIAGRAMAS DE FLUJO DE LAS SUBRUTINAS

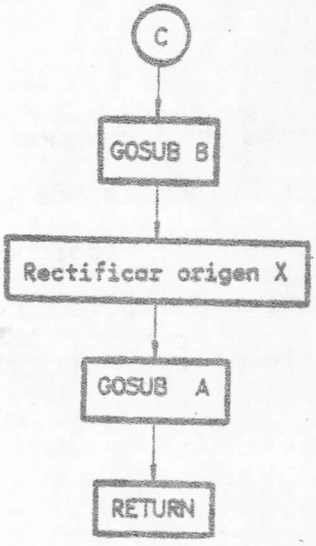
Subrutina A)



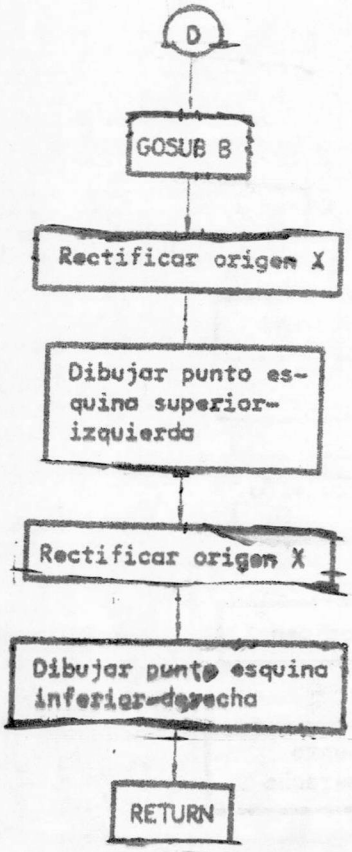
Subrutina B)



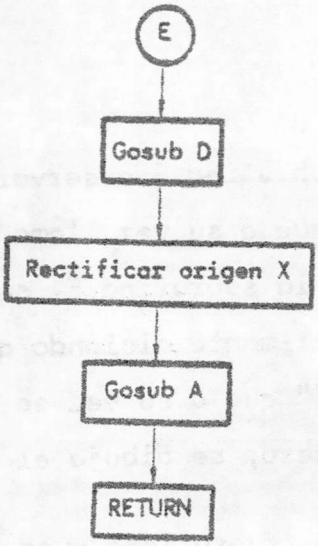
Subrutina C)



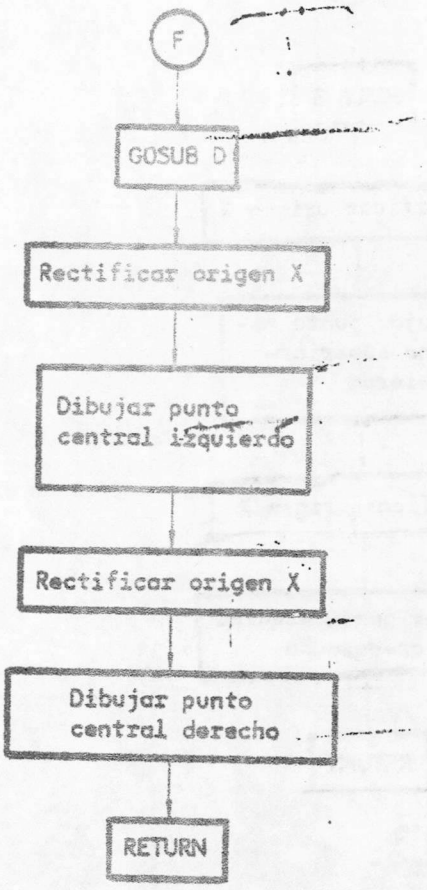
Subrutina D)



Subrutina E)



Subrutina F)



OBSERVACIONES :

Si se remite el lector a la subrutina E observará que esta subrutina llama a otra (la D), que a su vez llama a la B. Cuando de nuevo regresa el control a la subrutina E, se manda a la A. Esto se puede expresar más fácilmente diciendo que para dibujar un "5" dibujamos primero un "4" que a su vez se basa en el "2" y una vez finalizado este proceso, se dibuja el 1.

Como puede apreciarse todas las subrutinas quedan relacionadas entre sí, bajo formas similares a la explicada de manera que puedan ahorrarse el mayor número de instrucciones.

NOTA :

Se podían haber ahorrado algunos bloques de rectificación de origen X, pero se ha preferido seguir la norma de dibujar primero los puntos del dado de la parte izquierda para evitar, confusiones. Esto puede ser indicativo de que el ahorro de instrucciones no tiene por que tomarse como criterio rígido a la hora de elaborar un programa.

PROGRAMA

```
10 REM *JUEGO DE DADOS*
15 REM *FICHERO (3)*
20 CLS
25 Z=0
30 PRINT TAB(23);'JUEGO DE DADOS'
40 PRINT TAB(23);'==============='
50 PRINT
60 PRINT'REGLAS DEL JUEGO'
70 PRINT'----------------'
80 PRINT
90 PRINT'A) EN LA PRIMERA TIRADA ,SE GANA CON UNA SUMA DE 11 O 7 Y SE
100 PRINT'PIERDE CON 2,3 O 12 .'
110 PRINT
120 PRINT'B) EN LAS TIRADAS SIGUIENTES ,SE GANA IGUALANDO LA PRIMERA'
130 PRINT'TIRADA Y SE PIERDE CON UN 7 .'
140 PRINT
145 R=R+1
150 INPUT'PARA CONTINUAR PULSE UN (1)';C
160 IF C<>1 GOTO 150
170 CLS
180 PRINT@0,'                                                    ';
190 PRINT@0,' ';:INPUT'ADELANTE! TIRE LOS DADOS (PULSE UN 2)';T
200 IFT<>2 GOTO 180
210 A=RND(6):B=RND(6)
220 PRINT@280,'                    '
230 PRINT
235 PRINT
240 PRINT
245 PRINT
250 PRINT
255 PRINT
260 PRINT@280,'TIRADA NUMERO';Z+1
270 FOR X=14 TO 45
280 SET (X,15)
290 SET (X,30)
300 NEXT X
310 FOR X=82 TO 113
320 SET(X,15)
```

```
330 SET(X,30)
340 NEXT X
350 FOR Y=15 TO 30
360 SET(14,Y):SET(15,Y):SET(44,Y):SET(45,Y)
370 SET(82,Y):SET(83,Y):SET(112,Y):SET(113,Y)
380 NEXT Y
390 X=20
400 ON A GOSUB 1000,2000,3000,4000,5000,6000
410 X=88
420 ON B GOSUB 1000,2000,3000,4000,5000,6000
430 S=A+B
440 PRINT@473,"LA SUMA ES";S;
450 IF Z=0 GOTO 510
460 IF S=7 GOTO 580
470 IF S=P GOTO 560
480 PRINT@0,"SIGUIENTE TIRADA                                    "
485 FORI=1TO750:NEXTI
490 Z=Z+1
500 GOTO 180
510 IF(S=2)OR(S=3)OR(S=12) THEN 580
520 IF(S=7)OR(S=11) THEN 560
530 P=S
540 PRINT@206,"LA SUMA DE LA PRIMERA TIRADA VALE";P;
550 GOTO 480
560 PRINT@0,"LE FELICITO! HA GANADO                              "
570 GOTO 590
580 PRINT@0,"LO SIENTO! USTED PIERDE                             "
590 PRINT"LA PARTIDA HA FINALIZADO"
600 GOTO 600
999 END
1000 W=X+8
1010 FOR X=W TO W+3
1020 SET(X,22)
1030 SET(X,23)
1040 NEXTX
1050 RETURN
2000 W=X
2010 FOR X=W TO W+3
2020 SET(X,26)
2030 SET(X,27)
2040 NEXT X
2050 W=X+12
2060 FOR X=W TO W+3
2070 SET(X,18)
2080 SET(X,19)
2090 NEXT X
2100 RETURN
3000 GOSUB 2000
3010 X=X-20
3020 GOSUB 1000
3030 RETURN
4000 GOSUB 2000
4010 W=X-20
4020 FOR X=W TO W+3
4030 SET(X,18)
4040 SET(X,19)
4050 NEXT X
4060 W=X+12
4070 FOR X=W TO W+3
4080 SET(X,26)
4090 SET(X,27)
4100 NEXT X
4110 RETURN
5000 GOSUB 4000
5010 X=X-20
5020 GOSUB 1000
```

```
5030 RETURN
6000 GOSUB 4000
6010 W=X-20
6020 FOR X=W TO W+3
6030 SET(X,22)
6040 SET(X,23)
6050 NEXT X
6060 W=X+12
6070 FOR X=W TO W+3
6080 SET(X,22)
6090 SET(X,23)
6100 NEXT X
6110 RETURN
```

RELACION DIAGRAMA DE FLUJO - PROGRAMA

- Bloque de condiciones iniciales : línea 25.
- " " reglas del juego : líneas de la 3Ø a la 14Ø
- " lanzamiento de los dados : líneas de la 18Ø a la 21Ø
- " borrado de los dados anteriores : líneas de la 22Ø a la 255
- " número de tirada : línea 26Ø
- " dibujo de los dados : líneas de la 27Ø a la 42Ø
- Subrutinas : líneas de la 999 a la 411Ø
- Bloque de la suma : línea 43Ø y 44Ø
- " de decisión 1º tirada : línea 45Ø
- " " " S = 7 : línea 46Ø
- " " " S = P : línea 56Ø
- " " " S = 263612 : línea 51Ø
- " " " S = 7 ó 11 : línea 52Ø
- " memorización de la suma : línea 53Ø
- " suma 1º tirada : línea 54Ø
- " tirada siguiente : línea 48Ø
- " incremento del contador de tirada : línea 49Ø
- " gana el jugador : línea 56Ø
- " pierde el jugador : línea 58Ø

JUEGO DE LOS PALILLOS

NORMAS DEL JUEGO

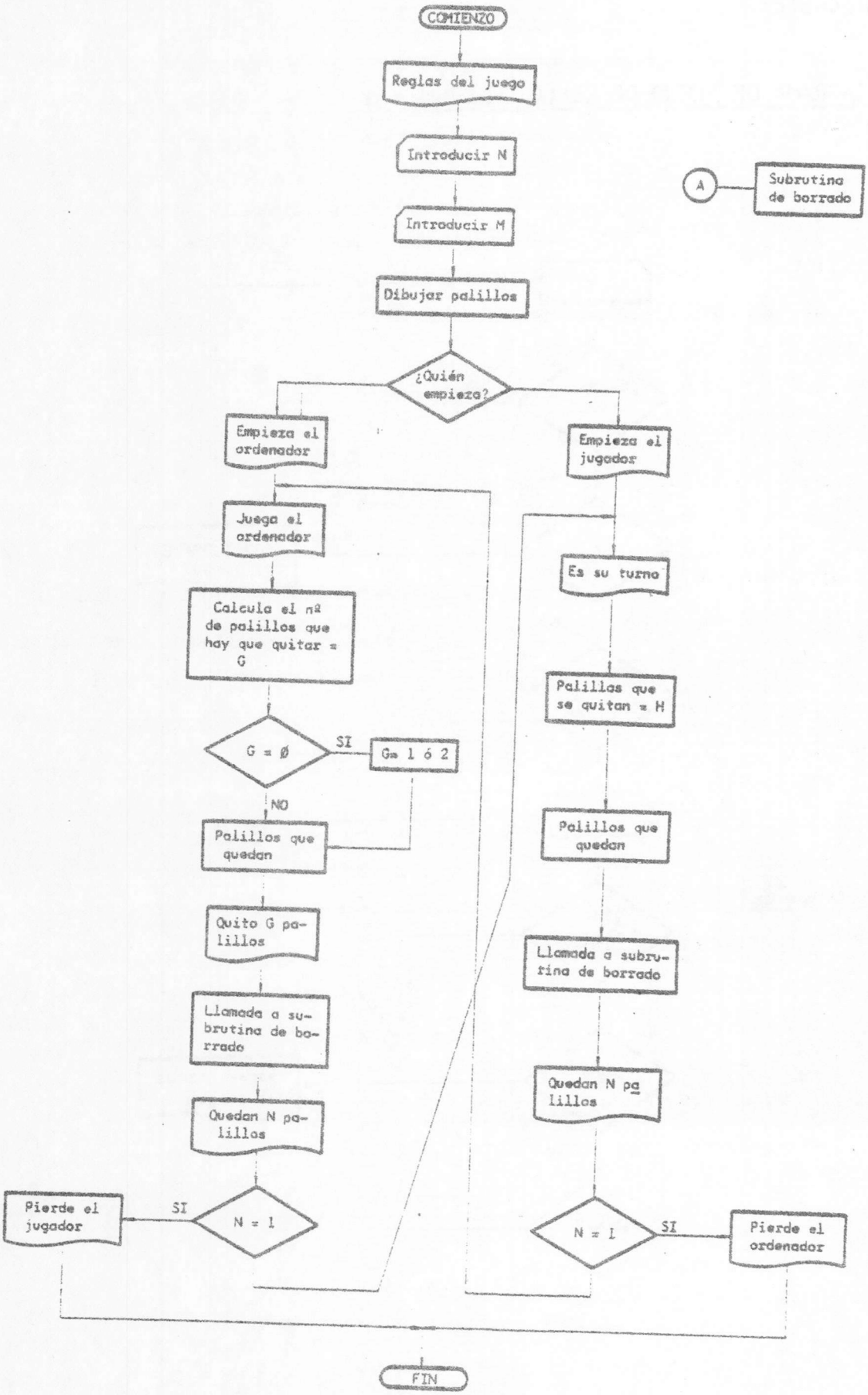
A) El jugador fija un número de palillos de partida (N) entre 12 y 21. También se elige el número máximo de palillos (M) ~~que puedan quitarse en cada tirada.~~ con la condición de que sean más de 2 y ~~menos~~ ó igual a N/4.

B) El juego consiste en ir quitando palillos alternati~~vamente, de manera que pierda el que quite el último~~.

El dibujo de los palillos en la pantalla se simula ~~por segmentos~~ verticales numerados en su parte superior. Asímismo en la parte superior de la imagen aparecen los mensajes adecuados. Cada vez que se quitan palillos, desaparecen segmentos de la pantalla de derecha a izquierda, restándose a la cantidad existente con anterioridad.

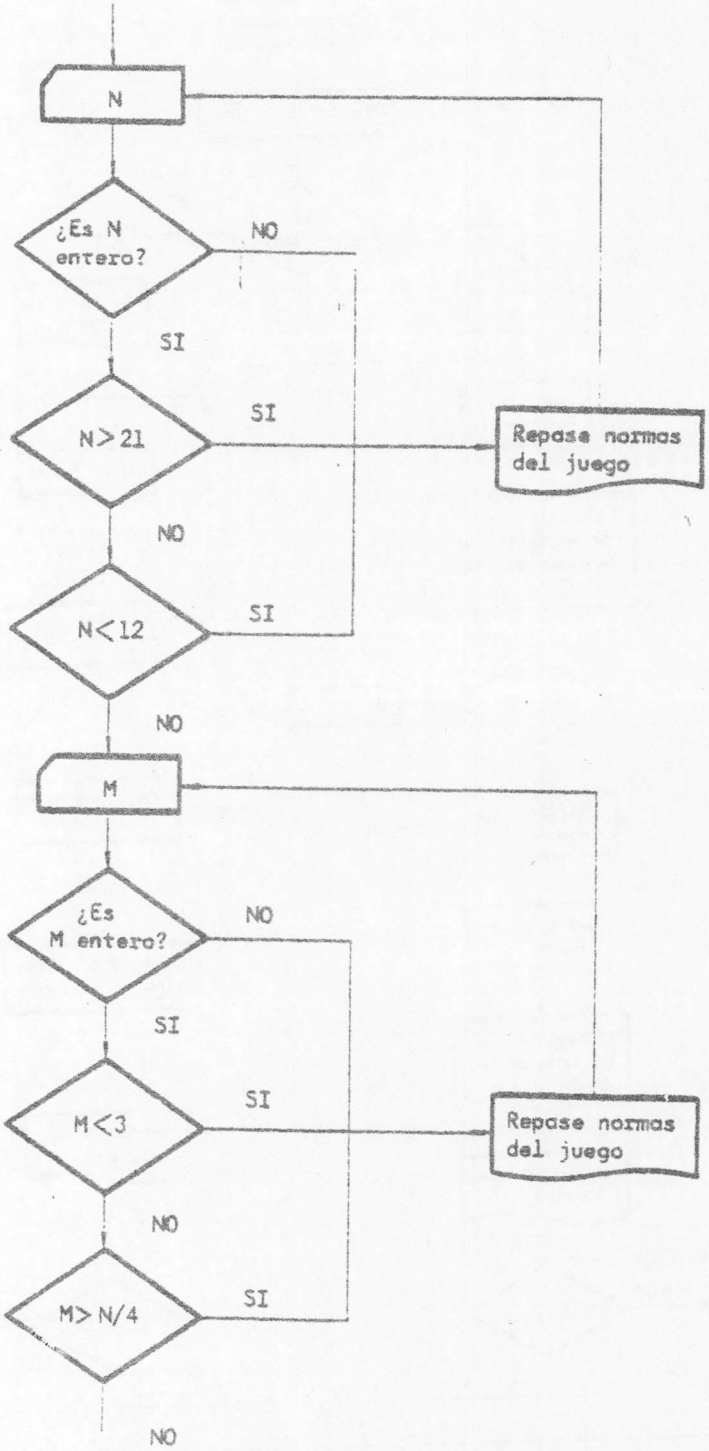
El diagrama de flujo general, describe las funciones que debe llevar a cabo el programa.

DIAGRAMA DE FLUJO GENERAL



Los bloques " introducir N y M" pueden desdoblarse en diagramas de flujo que expresan las condiciones de las normas de juego impuestas.

DIAGRAMA DE FLUJO DE ELECCION DE N y M



BLOQUE "CALCULO DE G"

Lo primero que hay que plantearse en este programa, es el sistema de cálculo de los palillos que debe quitar el ordenador, de manera que resulte más o menos "inteligente" su jugada. En realidad la dificultad de la mayoría de los programas de juegos reside precisamente en este punto, debido a que hay que salirse a menudo del terreno de la programación para internarse en campos que pueden ser menos conocidos. Así sucede, por ejemplo, con los programas relativos al juego de ajedrez.

El caso que aquí nos ocupa, utiliza una sencilla fórmula matemática para el cálculo de los palillos que debe quitar el ordenador (G). La fórmula es :

$$G = \left[\frac{N - 1}{M + 1} - INT\left(\frac{N - 1}{M + 1} \right) \right] (M + 1)$$

Si con unos determinados valores de N y M, G da un número distinto de cero, el ordenador gana la partida, pues en las tiradas siguientes se va a repetir esta circunstancia. Si, en cambio, G da cero, el ordenador debe quitar pocos palillos en espera del error del jugador.

Para explicar la fórmula mencionada, vamos a llamar tirada al hecho de quitar palillos cualquiera de los contendientes (ordenador, jugador) y jugada al conjunto tirada de ordenador-tirada de jugador.

El primer punto a tener en cuenta es que siempre se puede hacer que el número de palillos retirados en cada jugada, sea igual a (M + 1). Entonces el objetivo va a ser que partiendo de un número de palillos (N_1) y restándole G, se cumpla que el nuevo número de palillos que quede (No) sea, igual a K (M + 1) + 1, donde K es el número de jugadas al final de las cuales quedará un palillo.

1) $N_1 - G = No = K(M + 1) + 1$

de aquí se deduce :

2) $N_1 - 1 = K(M + 1) + G;$

3) $$\frac{N_1 - 1}{M + 1} = k + \frac{G}{M + 1}$$

4) $$G = \left(\frac{N_1 - 1}{M + 1} - K \right) (M + 1)$$

Ahora sólo resta demostrar que K es la parte entera de $\frac{N_1 - 1}{M + 1}$ cosa que vamos a razonar a continuación.

En la ecuación 3), se puede apreciar que $\frac{G}{M + 1}$ es menor que 1 porque siempre se cumple que G es menor que (M + 1); como por otra parte K es entero porque representa el número de jugadas, la única posibilidad para que se cumpla la ecuación es que K sea la parte entera de $\frac{N_1 - 1}{M + 1}$ y G la parte decimal.

Si la parte decimal es cero, es debido a que G es cero, lo que quiere decir, que el ordenador no tendría que quitar ningún palillo para quedar en situación óptima para ganar.

Como por otra parte, las reglas del juego indican que el mínimo número de palillos que hay que quitar como mínimo en cada tirada es 1, en esta situación, el jugador llevaría las de ganar.

PROGRAMA

```
10 REM * JUEGO DE LOS PALILLOS *
15 REM * FICHERO NUMERO 4 *
20 CLS
30 PRINTTAB(20);"JUEGO DE LOS PALILLOS"
40 PRINTTAB(20);"====================="
60 PRINT"REGLAS DEL JUEGO"
70 PRINT"----------------"
80 PRINT"A) EL JUGADOR ELIGE UN NUMERO DE PALILLOS (N) CON LA CONDICION"
90 PRINT"( 22>N>11 ) ."
100 PRINT"B) EL JUGADOR ELIGE EL NUMERO MAXIMO DE PALILLOS (M) ,QUE SE "
110 PRINT"PUEDEN QUITAR EN CADA TIRADA , CON LA CONDICION ( N/4>M>2 ).EL"
120 PRINT"MINIMO ES 1 ."
130 PRINT"C) TAMBIEN SE PUEDE ELEGIR QUIEN EMPIEZA A QUITAR PALILLOS"
140 PRINT"D) SE JUEGA ALTERNADAMENTE Y EL QUE COGE EL ULTIMO PIERDE ."
150 PRINT
200 INPUT"CON CUANTOS PALILLOS QUIERE JUGAR";N
210 IF (N-INT(N)=0)AND(N<22)AND(N>11) THEN 235
220 PRINT"REPASE LAS REGLAS DEL JUEGO Y VUELVA A ELEGIR (N)"
230 GOTO 200
235 PRINT
240 PRINT"CUAL ES EL NUMERO MAXIMO DE PALILLOS QUE SE PUEDEN ELEGIR EN"
250 PRINT"CADA TIRADA";:INPUT M
260 IF (M-INT(M)=0)AND(M>2)AND(M=<(N/4)) THEN 290
270 PRINT"REPASE LAS REGLAS DEL JUEGO Y VUELVA A ELEGIR (M)."
280 GOTO 240
290 CLS
300 Z=1
310 FOR X=3 TO N*6 STEP 6
320 PRINT@(445+3*Z),Z;
330 Z=Z+1
340 FOR Y=24 TO 38
350 SET(X,Y)
360 SET(X-1,Y)
370 NEXTY
380 NEXTX
400 PRINT@0,"                                                                    "
405 PRINT:PRINT
410 PRINT@0," ";:INPUT"QUIEN EMPIEZA? (1)-JUGADOR. (2)-ORDENADOR";E
420 IF (E<>2)AND(E<>1) GOTO 400
430 PRINT@0,"                                                                    "
435 PRINT:PRINT
440 IF E=2 GOTO 600
450 PRINT@0,"EMPIEZA USTED"
460 FORI=1TO750:NEXTI
470 PRINT@0," ";:INPUT"ES SU TURNO. CUANTOS PALILLOS QUITA";H
475 PRINT:PRINT
480 PRINT@0,"                                                                    "
485 PRINT:PRINT:PRINT
490 IF (H>0)AND(H<=M)AND(H-INT(H)=0) THEN 510
500 PRINT@0,"SE HA EQUIVOCADO AL MARCAR":GOTO 460
510 N=N-H
560 K=H
570 GOSUB 1000
580 PRINT@340,"QUEDAN";N;"PALILLOS    "
590 IF N=1 GOTO 760
595 GOTO 610
600 PRINT@0,"EMPIEZO YO"
610 FORI=1TO750:NEXTI
620 PRINT@0,"JUEGO YO      "
```

```
630 FORI=1TO500:NEXTI
640 G=(((N-1)/(M+1))-INT((N-1)/(M+1)))*(M+1)
650 IF G=0 GOTO 670
660 N=N-G:GOTO680
670 G=RND(2):GOTO 660
680 PRINT@0,"QUITO";G;"PALILLOS"
690 K=G
700 GOSUB 1000
710 PRINT@340,"QUEDAN";N;"PALILLOS"
720 IF N=1 GOTO 740
730 GOTO 470
740 PRINT@0,"LO SIENTO HA PERDIDO
750 GOTO 999
760 PRINT@0,"ENHORABUENA HA GANADO
999 GOTO 999
1000 FOR X=(6*(N+K)+1)TO(6*N+6) STEP-6
1010 FOR Y=21 TO 38
1020 RESET(X-3,Y)
1040 RESET(X-4,Y)
1050 RESET(X-5,Y)
1060 NEXT Y
1070 NEXT X
1080 RETURN
```

RELACION DIAGRAMA DE FLUJO - PROGRAMA

Bloque "reglas del juego"	Líneas de la 3Ø a la 15Ø
" "introducir N"	" " " 2ØØ a la 235
" " " M"	" " " 24Ø " " 28Ø
" "dibujar palillos"	" " " 3ØØ " " 38Ø
" "decidir, quién empieza"	" " " 4ØØ " " 435
" de decisión "quién empieza	línea 44Ø
" "empieza el jugador"	líneas 45Ø y 46Ø
" "palillos que se quitan"	líneas de la 47Ø a la 5ØØ
" "palillos que quedan"	línea 51Ø
" llamada a "subrutina de borrado"	línea 57Ø
" "quedan N palillos"	línea 58Ø
" de decisión "N = 1"	línea 59Ø
" "pierde el ordenador"	línea 76Ø
" "empieza el ordenador"	líneas 6ØØ y 61Ø
" "juega el ordenador"	líneas 62Ø y 63Ø
" "calcula G"	línea 64Ø
" de decisión "G = Ø"	línea 65Ø

Bloque	"G = 1 ó 2"	línea 67Ø
"	"palillos que quedan"	línea 66Ø
"	"quito G palillos"	línea 68Ø
"	"Gosub (A) "	línea 7ØØ
"	"quedan N palillos"	línea 71Ø
"	de decisión "N = 1"	línea 72Ø
"	"pierde el jugador"	línea 74Ø
"	subrutina "borrar palillos"	líneas de la 1ØØØ a la 1Ø8Ø

JUEGO DE ALUNIZAJE

NORMAS DEL JUEGO

A) En este juego se trata de simular el alunizaje de una nave espacial. La acción de la gravedad del planeta, ejerce una aceleración que puede ser contrarrestada por la acción del combustible al quemarse.

B) Como punto de partida se toman valores aleatorios (dentro de un margen), de velocidad, altura y cantidad de combustible. El juego consiste entonces en tomar tierra con una velocidad lo más próxima a cero, sabiendo que el impulso debido a la combustión se produce cada segundo. Por esta razón, al final de cada segundo, se imprimen los nuevos valores de las tres variables.

C) En el último segundo en que se toma tierra, se calcula la velocidad de choque que determinará la calificación del alunizaje; si la velocidad de choque supera un cierto valor, la nave se rompe.

D) En ningún momento se debe dejar subir a la nave. Hay que tener esto en cuenta para no permitir en el programa un cambio de signo de la velocidad.

En la pantalla de video se dibuja la nave con los datos de las variables a su lado izquierdo. En la parte superior se imprimen los mensajes relativos a las normas del juego.

Antes de abordar el programa parece conveniente comenzar por un estudio del movimiento de la nave.

ESTUDIO DEL MOVIMIENTO

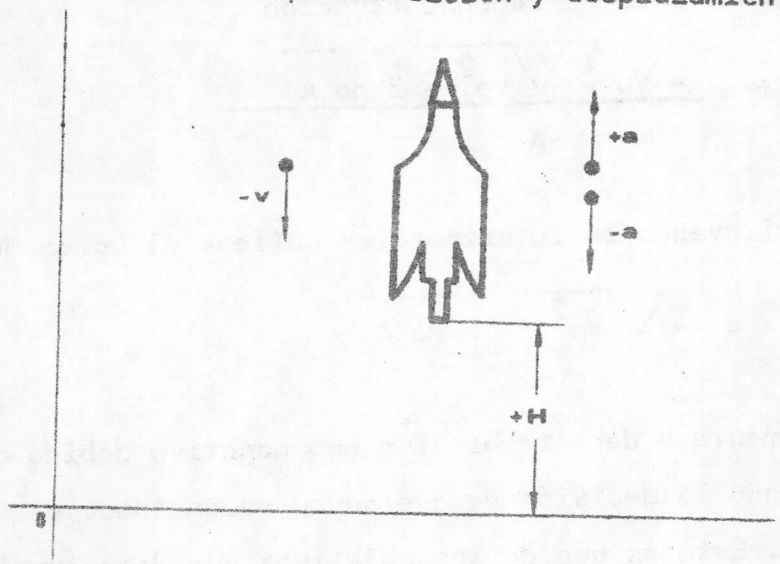
Las ecuaciones generales del movimiento uniformemente acelerado son :

$$V = Vo + a.t$$

$$H = Ho + Vo.t + \frac{1}{2} a.t^2$$

V : velocidad final
Vo : velocidad inicial
a : aceleración
t : tiempo transcurrido
H : altura final
Ho : altura inicial

Por convenio elegiremos como negativo el sentido hacia abajo de los vectores velocidad, aceleración y desplazamiento (altura)



La aceleración, puede tener los dos sentidos, en tanto que la velocidad y la altura sólo negativo y positivo respectivamente.

La expresión de la aceleración en este caso particular viene dada por la ecuación :

$$A = 2Q - 5$$

en donde Q es el combustible que se elige en cada segundo, por lo tanto A es constante para ese segundo.

Particularizando para t = 1 las ecuaciones generales del movimiento se convierten en :

$$V = Vo + A$$

$$H = Ho + Vo + A/2$$

En cambio, el cálculo de la velocidad de choque se realiza en el intervalo del último segundo, por eso t es distinto de 1. La velocidad toma la expresión que seguidamente deduciremos :

1) $V = Vo + A$

2) $0 = Ho + Vo.t + 1/2 \ A.t^2$

Despejando t en la segunda ecuación :

$$t = \frac{-Vo \pm \sqrt{Vo^2 - 2.Ho.A}}{A}$$

y sustituyendo en la primera se obtiene el valor de V :

$$V = \pm\sqrt{Vo^2 - 2\ Ho\ A}$$

Lógicamente V debe tomar el signo negativo debido a que se ha tomado la decisión de que nunca pueda tomar el valor positivo. Este es uno de los objetivos que debe cumplir el programa.

DIAGRAMA DE FLUJO GENERAL

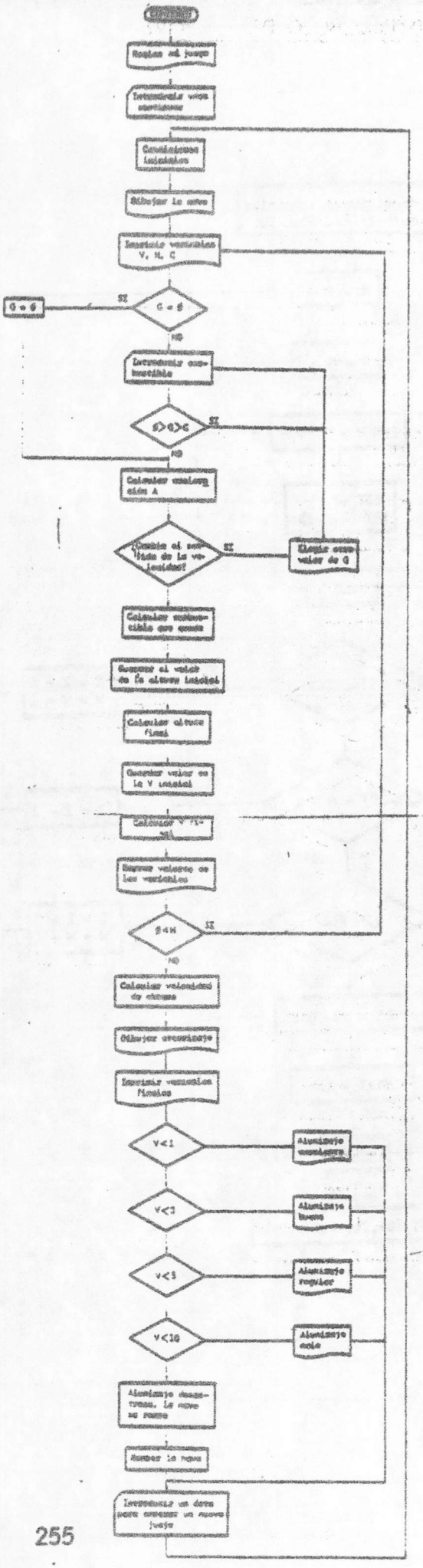
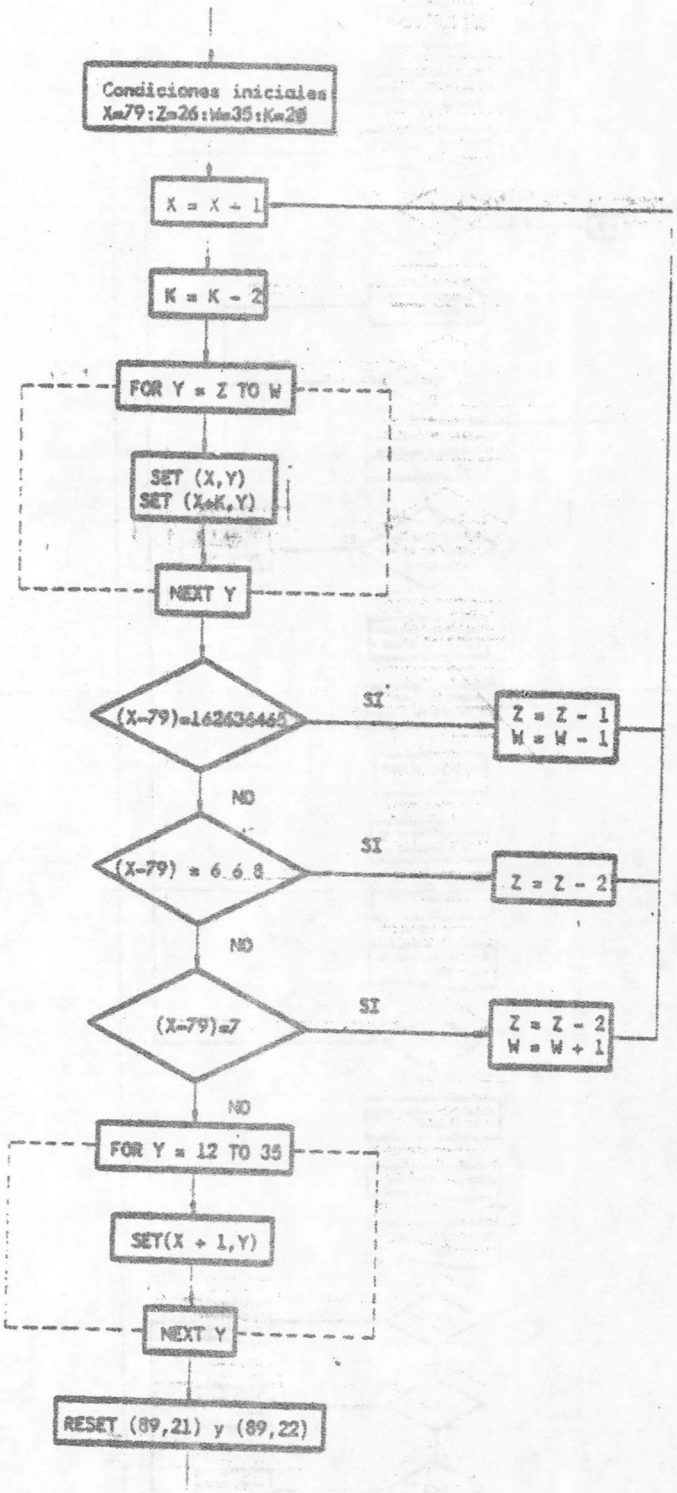


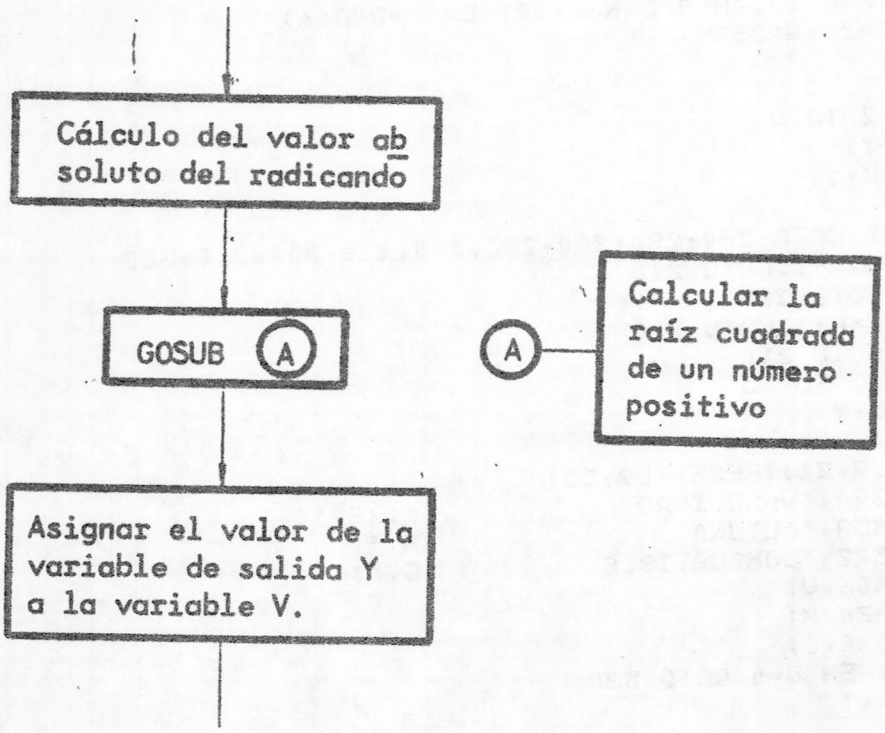
DIAGRAMA DE FLUJO DEL DIBUJO DE LA NAVE



NOTA :

Para seguir el diagrama de flujo del dibujo de la nave, se recomienda ayudarse de la hoja de trabajo para la pantalla de video. También ayuda a la comprensión de los bloques, imprimir y borrar variables.

DIAGRAMA DE FLUJO DEL CALCULO DE LA VELOCIDAD DE CHOQUE



PROGRAMA

```
10 REM * JUEGO DEL ALUNIZAJE *
15 REM * FICHERO NUMERO 5 *
18 CLS
20 PRINT TAB(20);"JUEGO DEL ALUNIZAJE"
30 PRINT TAB(20);"====================="
40 PRINT
50 PRINT"NORMAS DEL JUEGO"
60 PRINT"----------------"
70 PRINT
80 PRINT"A) EN ESTE JUEGO ,USTED SE INTRODUCE DENTRO DE UN COHETE EN EL"
90 PRINT"MOMENTO DEL ALUNIZAJE.LOS MOTORES ESTAN BAJO SU MANDO.LA QUEMA"
100 PRINT"DE COMBUSTIBLE OCASIONA UNA ACELERACION QUE SE OPONE A LA CAIDA."
120 PRINT"B) EL OBJETIVO ES APLICAR ENCENDIDOS EN CADA SEGUNDO, PARA QUE"
130 PRINT"SE LLEGUE A LA SUPERFICIE LUNAR CON UNA VELOCIDAD LO MAS PRO-"
135 PRINT"XIMA A CERO."
140 PRINT
150 INPUT "PARA CONTINUAR PULSE UN (1)";S
160 IF S<>1 GOTO 150
170 CLS
180 V=-55+RND(10):H=475+RND(50):C=57+RND(6)
200 X=79:Z=26:W=35:K=20
210 X=X+1
220 K=K-2
230 FOR Y=Z TO W
240 SET(X,Y)
250 SET(X+K,Y)
260 NEXT Y
270 ON X-79 GOTO 280,280,280,280,280,290,300,310,320
280 Z=Z-1:W=W-1:GOTO 210
290 Z=Z-2:GOTO 210
300 Z=Z-2:W=W+1:GOTO 210
310 Z=Z-2:GOTO 210
320 FOR Y=12 TO 35
330 SET(X+1,Y)
340 NEXT Y
350 RESET(89,21):RESET(89,22)
400 PRINT@394,"VELOCIDAD                    M./S.";
410 PRINT@458,"ALTURA                       M.";
420 PRINT@522,"COMBUSTIBLE                  G.";
430 PRINT@406,V;
440 PRINT@470,H;
450 PRINT@534,C;
460 IFC=0 THEN Q=0:GOTO 530
480 PRINT@0,"
490 PRINT
500 PRINT@0," ";
510 PRINT"QUE CANTIDAD DE COMBUSTIBLE DESEA QUEMAR";
520 INPUT Q
530 PRINT@0,"
540 PRINT:PRINT
550 IF Q<0 GOTO 480
555 IF Q>C GOTO 480
560 A=2*Q-5
570 IFV+A<=0 GOTO 600
580 PRINT@0,"ELIJA MENOS COMBUSTIBLE."
590 PRINT"CON EL COMBUSTIBLE QUE ACABA DE ELEGIR, EL COHETE SUBIRIA."
595 FOR I=1TO1000:NEXTI:GOTO 480
600 C=C-Q:C=INT(C*10000)/10000
610 L=H
```

```
620 H=H+A/2+V:H=INT(H*1000)/1000
630 N=V
640 V=V+A:V=INT(V*1000)/1000
642 PRINT@406,"          ";
644 PRINT@470,"          ";
646 PRINT@534,"          ";
650 IF H>0 GOTO 430
670 X=ABS(2*A*L-N*N)
680 GOSUB 30030
690 V=Y:V=INT(V*1000)/1000
700 Y=46
710 FOR X=50 TO 120
720 SET(X,Y)
730 RESET(X,Y+1)
740 NEXT X
750 IFY=36 GOTO 800
760 Y=Y-1:GOTO 710
800 PRINT@386,"VELOCIDAD DE CHOQUE";
820 PRINT@406,-V;
840 PRINT@471,"00";
860 PRINT@534,C;
900 IF V<1 PRINT@0,"ALUNIZAJE EXCELENTE":GOTO 990
910 IF V<3 PRINT@0,"ALUNIZAJE BUENO":GOTO 990
920 IF V<5 PRINT@0,"ALUNIZAJE REGULAR":GOTO 990
930 IF V<10 PRINT@0,"ALUNIZAJE MAS BIEN MALO":GOTO 990
940 PRINT@0,"ALUNIZAJE DESASTROSO.SE QUEDA PARA SIEMPRE EN LA LUNA"
950 X=78:Y=35
960 X=X+1:Y=Y-1
970 RESET(X,Y):RESET(X,Y-1)
980 IF X<98 GOTO 960
990 FOR I=1 TO 2000:NEXT I
992 PRINT@0,"
995 PRINT@0," ";:INPUT"PULSANDO UN (1) SE COMIENZA DE NUEVO";S                    ";
1000 IF S=1 GOTO 170
1010 GOTO 992
30030 IF X=0 THEN Y=0:RETURN
30040 IF X>0 THEN 30060
30050 PRINT"RAIZ CUADRADA DE UN NUMERO NEGATIVO":STOP
30060 Y=X*.5:Z=0
30070 W=(X/Y-Y)*.5
30080 IF(W=0)OR(W=Z) THEN RETURN
30090 Y=Y+W:Z=W:GOTO 30070
```

RELACION DIAGRAMA DE FLUJO - PROGRAMA

Bloque	"Reglas del juego"	Líneas de la 2Ø a la 14Ø
"	"Introducir un 1"	líneas 15Ø y 16Ø
"	"Condiciones iniciales"	línea 18Ø
"	"Dibujar nave"	líneas de la 2ØØ a la 35Ø
"	"imprimir variables"	líneas " " 4ØØ " " 45Ø
"	de condición " C= Ø"	línea 46Ø
"	"introducir combustible"	líneas de la 49Ø a la 54Ø
"	de condición "Ø>Q>C"	líneas 55Ø y 555
"	"calcular aceleración"	línea 56Ø
"	de condición "cambia sentido de V"	línea 57Ø
"	"elegir otro valor de Q"	líneas de la 58Ø a la 595
"	"calcular combustible"	línea 6ØØ
"	"guarda H inicial"	línea 61Ø
"	"calcular H final"	línea 62Ø
"	"guardar V inicial"	línea 63Ø
"	"calcular V final"	línea 64Ø
"	"borrar valores de las variables"	líneas de la 642 a la 646
"	de condición "H> Ø"	línea 65Ø
"	"calcular V de choque"	líneas de la 67Ø a la 69Ø
"	"dibujar aterrizaje"	líneas " " 7ØØ " " 76Ø
"	"imprimir variables finales"	líneas " " 8ØØ " " 86Ø
"	"calificación resultados"	líneas " " 9ØØ " " 94Ø
"	"romper la nave "	líneas " " 95Ø " " 99Ø
"	"pulsando (1) nuevo juego"	líneas " " 992 " " 1Ø1Ø
"	"Subrutina"	líneas " " 3ØØ3Ø a la 3ØØ9Ø

JUEGO DE MUERTOS Y HERIDOS

REGLAS DEL JUEGO

El jugador juega contra el ordenador intentando adivinar un número de 3 cifras siguiendo las reglas siguientes :

A) El jugador elige quién comienza a jugar (el ordenador ó él mismo). Para no dar ventajas a nadie se debería escoger de forma alternada el inicio de cada juego.

B) Tanto el ordenador como el jugador eligen un número de 3 cifras, con las características de que el número no empiece por cero y no se repita ninguna de sus cifras.

C) El ordenador y el jugador escriben alternativamente números con el objetivo de adivinar el que secretamente guarda su contrario. Al cabo de cada tirada cada oponente debe informar del número de muertos y heridos conseguidos (se entiende por muerto la cifra acertada en su valor y posición siendo herido, si sólamente se acierta el valor).

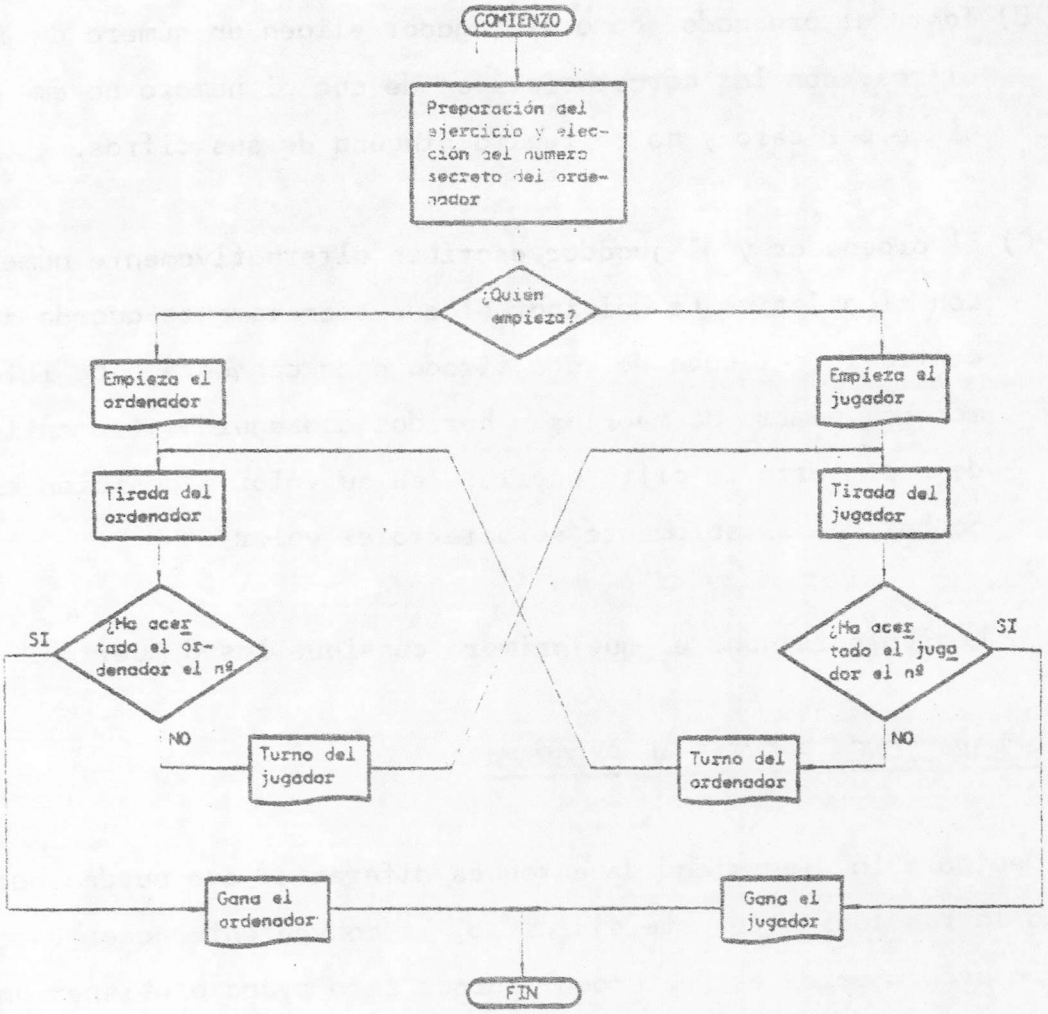
D) Lógicamente gana el que primero consigue los 3 muertos.

ACLARACIONES PREVIAS AL PROGRAMA

Debido a la diversidad de enfoques diferentes que pueden darse a la resolución de este ejercicio, es conveniente hacer uso de la programación estructurada, porque esto ayuda a obtener una visión global del programa y además permite realizar modificaciones con relativa sencillez como se verá al final del ejercicio.

Por otra parte, hemos de decir que vamos a utilizar una forma muy particular de impresión de resultados en la pantalla de video. Con objeto de facilitar al jugador la información de muertos y heridos conseguidos en cada tirada, así como el control de las mismas, se lleva a cabo una impresión en forma tabulada haciendo uso de las funciones gráficas conocidas.

DIAGRAMA DE FLUJO GENERAL



Este programa utiliza como subrutinas :

SUBRUTINA A

(A) — Generar un número aleatorio de 3 cifras diferentes que no empiece por cero

SUBRUTINA B

(B) — Calcular muertos y heridos

La subrutina B hace una llamada a otra subrutina.

Seguidamente se van a desarrollar los bloques : Preparación del ejercicio, Tirada del jugador, Tirada del ordenador.

DIAGRAMA DE FLUJO "PREPARACION DEL EJERCICIO"

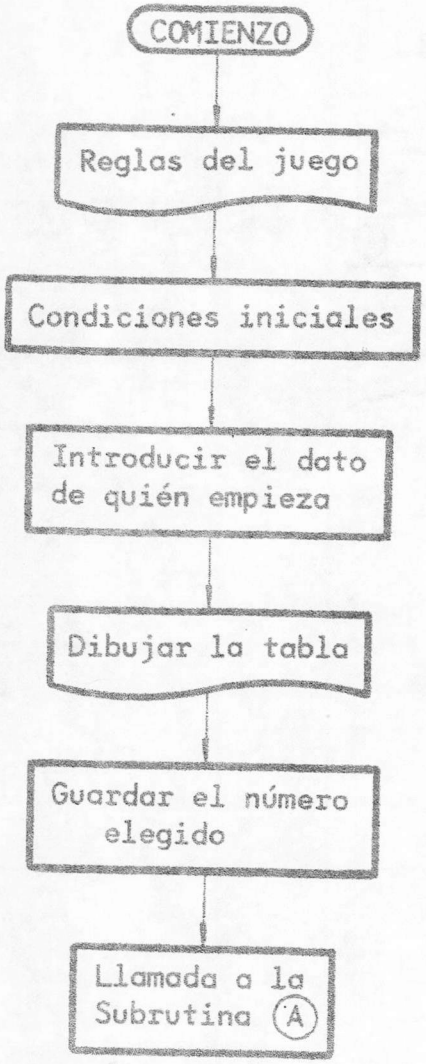


DIAGRAMA DE FLUJO DEL BLOQUE "TIRADA DEL JUGADOR"

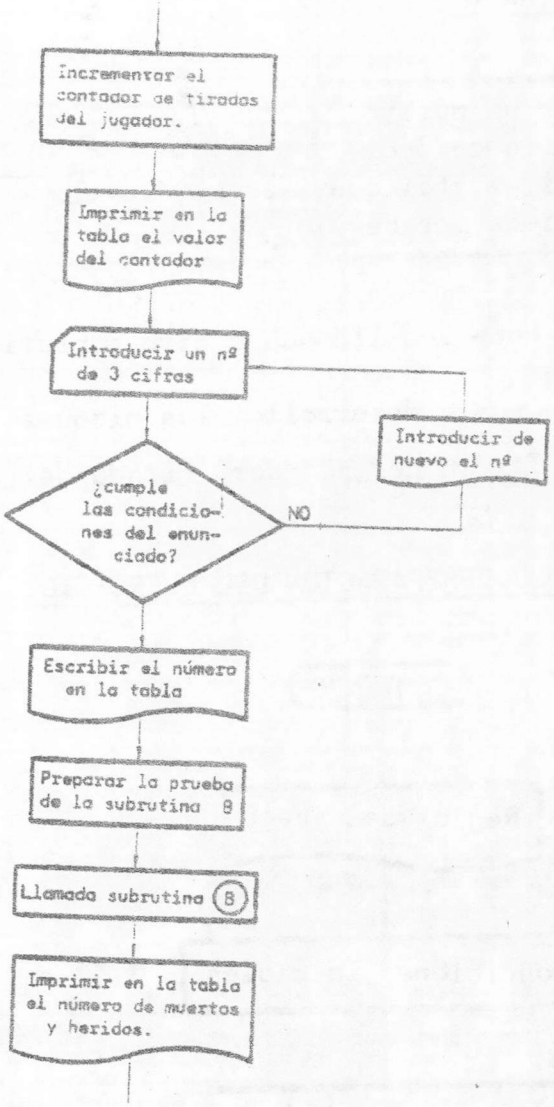
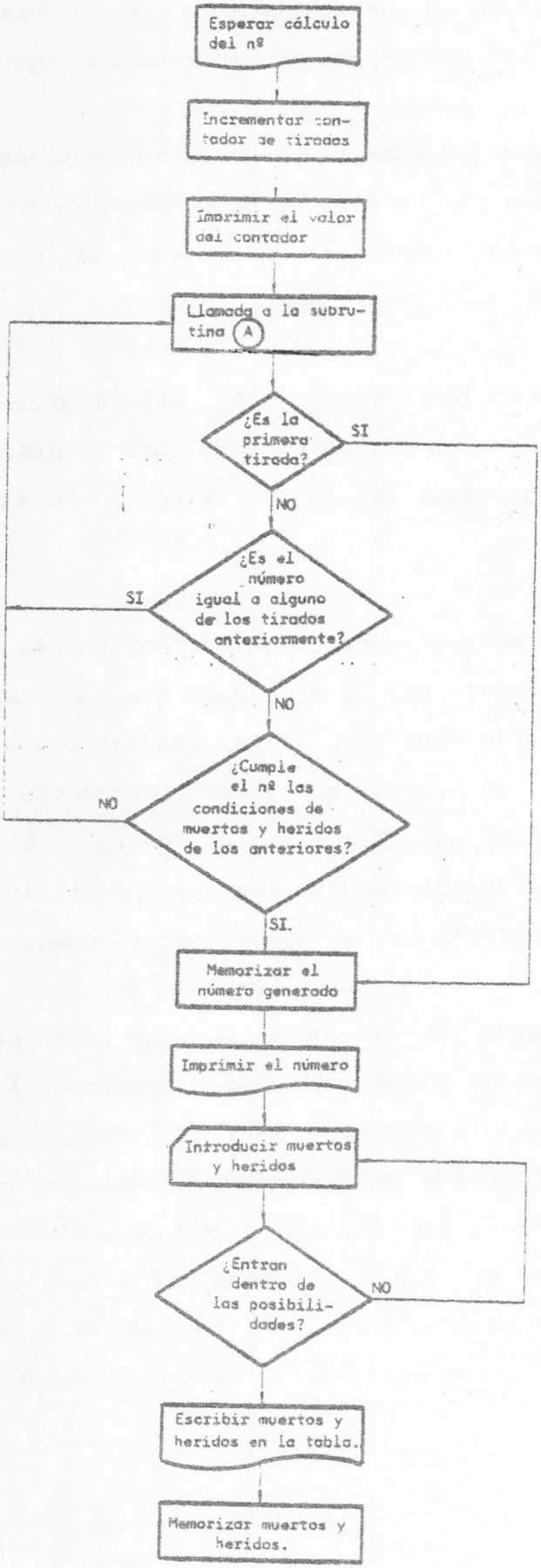


DIAGRAMA DE FLUJO DEL BLOQUE "TIRADA DEL ORDENADOR"



Este diagrama de flujo explica el método que se ha seguido en el cálculo del número que tira el ordenador y puede resumirse diciendo :

- El ordenador genera un número aleatorio en la subrutina A que no empiece por cero y que no tenga cifras iguales.

- La primera prueba que pasa es la pregunta de si se trata del primer número. Si es así, el ordenador lo da como válido, ya que en la primera tirada se puede decir cualquier número.

- Si no se sobrepasa la primera prueba, quiere decir que no se trata de la primera tirada en cuyo caso se pregunta si el número al azar generado por la subrutina A ha sido elegido anteriormente.

- Si el número no ha sido elegido anteriormente, el ordenador supone por un momento que es el número que ha pensado el jugador, en cuyo caso debe cumplir las condiciones de muertos y heridos de los anteriores que tiene almacenados (función realizada por el bloque "Memorizar el número"). Si pasa la prueba, tiene la posibilidad de acertar, posibilidad que está en relación directa con el aumento de tiradas.

Como puede apreciarse este método es un poco lento debido a que la subrutina puede repetir el número generado. Además si se introducen valores erróneos de muertos y heridos por el jugador, el ordenador queda en un ciclo loco del que no puede salir. Más adelante se estudiará la forma de evitar estos inconvenientes. Si se estudia el programa tal y como está, es debido a que se desea resaltar la utilidad del diagrama de flujo a la hora de efectuar modificaciones dentro de dicho programa.

Al mismo tiempo, supone un ejemplo, como se verá más adelante, de la pluralidad de soluciones o enfoques diferentes que ofrece la programación, para resolver un determinado problema.

PROGRAMA

```
10 REM *JUEGO DE MUERTOS Y HERIDOS*
15 REM *FICHERO NUMER 2*
16 DIM A(110)
20 CLS
30 PRINT TAB(18);"JUEGO DE MUERTOS Y HERIDOS"
40 PRINT TAB(18);"==========================="
50 PRINT "REGLAS DEL JUEGO"
60 PRINT "-----------------"
70 PRINT"A) EL ORDENADOR Y EL JUGADOR ELIGEN UN NUMERO DE 3 CIFRAS DIS-"
80 PRINT"TINTAS QUE NO EMPIECE POR CERO."
85 PRINT
90 PRINT"B) DE MANERA ALTERNATIVA SE VAN TIRANDO NUMEROS E INFORMANDO "
100 PRINT"DEL NUMERO DE MUERTOS Y HERIDOS ACERTADOS."
105 PRINT
110 PRINT"C) GANA EL QUE ANTES CONSIGUE LOS 3 MUERTOS."
120 PRINT
130 T=0:J=0
140 PRINT"QUIEN EMPIEZA?"
150 PRINT"SI SE PULSA UN 1 (EL JUGADOR) Y CON UN 2 (EL ORDENADOR)";
160 INPUT E
170 IF (E<>1)AND(E<>2) THEN 120
180 CLS
200 FOR X=0 TO 127
210 SET(X,13)
220 SET(X,18)
230 SET(X,47)
240 NEXT X
250 FOR Y=13 TO 47
260 SET(0,Y)
270 SET(15,Y)
280 SET(43,Y)
290 SET(53,Y)
300 SET(63,Y)
310 SET(64,Y)
320 SET(79,Y)
330 SET(107,Y)
340 SET(117,Y)
350 SET(127,Y)
360 NEXT Y
370 PRINT@321,"TIRADA";
380 PRINT@329,"N.JUGADOR";
390 PRINT@343,"M.";
400 PRINT@348,"H.";
410 PRINT@353,"TIRADA";
420 PRINT@361,"N.ORDENADOR";
430 PRINT@375,"M.";
440 PRINT@380,"H.";
450 PRINT@0," ";
460 GOSUB 2000
500 A(5)=A(1):A(6)=A(2):A(7)=A(3)
510 IF E=1 GOTO 550
520 IF E=2 GOTO 800
550 PRINT@0,"EMPIEZA USTED
560 FORI=1TO500:NEXTI
570 J=J+1:PRINT@(386+J*64),J;
580 PRINT@0,"INTRODUZCA UN NUMERO DE 3 CIFRAS VALIDO"
590 FORI=1TO1000:NEXTI
600 PRINT@0,"
610 PRINT@0," ";:INPUT"PRIMERA CIFRA";V
620 PRINT@0," ";:INPUT"SEGUNDA CIFRA";X
630 PRINT@0," ";:INPUT"TERCERA CIFRA";Y
640 PRINT@0,"
650 IF(V>9)OR(X>9)OR(Y>9) THEN 690
660 IF((V-INT(V))<>0)OR((X-INT(X))<>0)OR((Y-INT(Y))<>0) THEN 690
```

```
670 IF(V<1)OR(V=X)OR(V=Y)OR(X=Y)OR(X<0)OR(Y<0) THEN 690
680 GOTO 710
690 PRINT@0,"EL NUMERO NO ES VALIDO. ESCRIVALE DE NUEVO"
700 GOTO 590
710 PRINT@(394+J*64),V;X;Y;
720 A(1)=A(5):A(2)=A(6):A(3)=A(7)
730 GOSUB 3000
740 PRINT@(407+J*64),M;
750 PRINT@(412+J*64),H;
760 IF M=3 GOTO 1200
780 PRINT@0,"NO HA ACERTADO Y AHORA ME TOCA A MI"
790 GOTO 820
800 PRINT@0,"
810 PRINT@0,"EMPIEZO YO"
820 FORI=1TO750:NEXTI
830 PRINT@0,"
840 PRINT@0,"ESPERE A QUE CALCULE EL NUMERO"
850 FORI=1TO500:NEXTI
860 T=T+1
870 PRINT@(418+T*64),T;
880 GOSUB 2000
890 IFT=1 GOTO 990
900 FOR K=1 TO T-1
910 IF(A(10*K+1)=A(1))AND(A(10*K+2)=A(2))AND(A(10*K+3)=A(3))GOTO880
920 NEXT K
930 FOR K=1 TO T-1
940 V=A(10*K+1):X=A(10*K+2):Y=A(10*K+3)
950 GOSUB 3000
960 IF(M=A(10*K+5))AND(H=A(10*K+6)) GOTO 980
970 GOTO 880
980 NEXT K
990 A(10*T+1)=A(1):A(10*T+2)=A(2):A(10*T+3)=A(3)
1000 PRINT@0,"MI NUMERO ES EL :";A(1);A(2);A(3);"
1010 FORI=1TO750:NEXTI
1020 PRINT@(426+64*T),A(1);A(2);A(3);
1030 PRINT@0,"INTRODUZCA MUERTOS Y HERIDOS";
1040 FORI=1TO750:NEXTI
1050 PRINT@0,"
1060 PRINT@0," ";:INPUT"NUMERO DE MUERTOS";M
1070 PRINT@0,"
1080 PRINT@0," ";:INPUT"NUMERO DE HERIDOS";H
1090 IF(M>3)OR(H>3)OR(M+H>3)OR(M-INT(M)<>0)OR(H-INT(H)<>0) GOTO 1030
1100 IF(M<0)OR(H<0)OR(M=2)AND(H=1) GOTO 1030
1110 PRINT@0,"
1120 PRINT@(439+64*T),M;
1130 PRINT@(444+64*T),H;
1140 A(10*T+5)=M:A(10*T+6)=H
1150 IF M=3 THEN 1180
1160 PRINT@0,"ES SU TURNO";
1170 GOTO 560
1180 PRINT@0,"HE GANADO. MI NUMERO ERA EL";A(5);A(6);A(7)
1190 GOTO 1999
1200 PRINT@0,"ENHORBUENA, HA GANADO USTED"
1999 GOTO 1999
2000 FOR I=1 TO 3
2010 A(I)=RND(10)-1
2020 NEXT I
2030 IF(A(1)=0)OR(A(1)=A(2))OR(A(1)=A(3))OR(A(2)=A(3)) THEN 2000
2040 RETURN
3000 M=0:H=0
3010 IF V=A(1) GOSUB 4000
3020 IF X=A(1) GOSUB 5000
3030 IF Y=A(1) GOSUB 5000
3040 IF V=A(2) GOSUB 5000
3050 IF X=A(2) GOSUB 4000
3060 IF Y=A(2) GOSUB 5000
3070 IF V=A(3) GOSUB 5000
3080 IF X=A(3) GOSUB 5000
3090 IF Y=A(3) GOSUB 4000
3100 RETURN
4000 M=M+1
4010 RETURN
5000 H=H+1
5010 RETURN
```

RELACION DIAGRAMA DE FLUJO - PROGRAMA

Bloque	"Reglas del juego"	líneas de la 1Ø a la 12Ø
"	"Condiciones iniciales"	línea 13Ø
"	"Introducir quién empieza"	líneas de la 14Ø a la 17Ø
"	"Dibujar tabla"	líneas de la 2ØØ a la 46Ø
"	"Guardar el número elegido"	línea 5ØØ
"	de condición "Quién empieza"	líneas 51Ø y 52Ø
"	"Empieza el jugador"	líneas 55Ø y 56Ø
"	"Incrementar el contador de tiradas"	línea 57Ø
"	"Imprimir el valor del contador"	línea 57Ø
"	"Introducir un nº de 3 cifras"	líneas de la 58Ø a la 64Ø
"	de condición "Cumple condiciones del enunciado"	líneas de la 65Ø a la 67Ø
"	"Introducir de nuevo el número"	línea 69Ø
"	"Dibujar el nº en la tabla"	línea 71Ø
"	"Preparar prueba de la subrutina (B)"	línea 72Ø
"	"Llamada a la subrutina (B)"	línea 73Ø
"	"Imprimir en la tabla muertos y heridos"	líneas 74Ø y 75Ø
"	de condición "Ha acertado el jugador el número"	línea 76Ø
"	"Turno del ordenador"	línea 78Ø
"	"Empieza el ordenador"	líneas de la 8ØØ a la 83Ø
"	"Esperar cálculo del número"	líneas 84Ø y 85Ø
"	"Incrementar el contador de tirada"	línea 86Ø
"	"Imprimir el valor del contador"	línea 87Ø
"	"Llamada subrutina (A)"	línea 88Ø
"	de condición "¿Es la 1ª. tirada"?	línea 89Ø
"	de condición "¿Es el número igual a alguno de los anteriores?"	líneas de la 9ØØ a la 92Ø

Bloque	de condición "¿Cumple el número las condiciones de muertos y heridos de los anteriores?"	líneas de la 93Ø a la 98Ø
"	"Memorizar el número generado"	línea 99Ø
"	"Imprimir el número"	líneas de la 1ØØØ a la 1Ø2Ø
"	"Introducir muertos y heridos"	líneas de la 1Ø3Ø a la 1Ø8Ø
"	de condición "Entran dentro de las posibilidades"	líneas 1Ø9Ø y 11ØØ
"	"Escribir muertos y heridos en la tabla"	líneas 112Ø y 113Ø
"	"Memorizar muertos y heridos"	línea 114Ø
"	de condición "Ha acertado el ordenador el número"	línea 115Ø
"	"Turno del jugador"	línea 116Ø
"	"Gana el ordenador"	línea 118Ø
"	"Gana el jugador"	línea 12ØØ
"	"Subrutina Ⓐ "	líneas de la 2ØØØ a la 2Ø4Ø
"	"Subrutina Ⓑ "	líneas de la 3ØØØ a la 5Ø1Ø

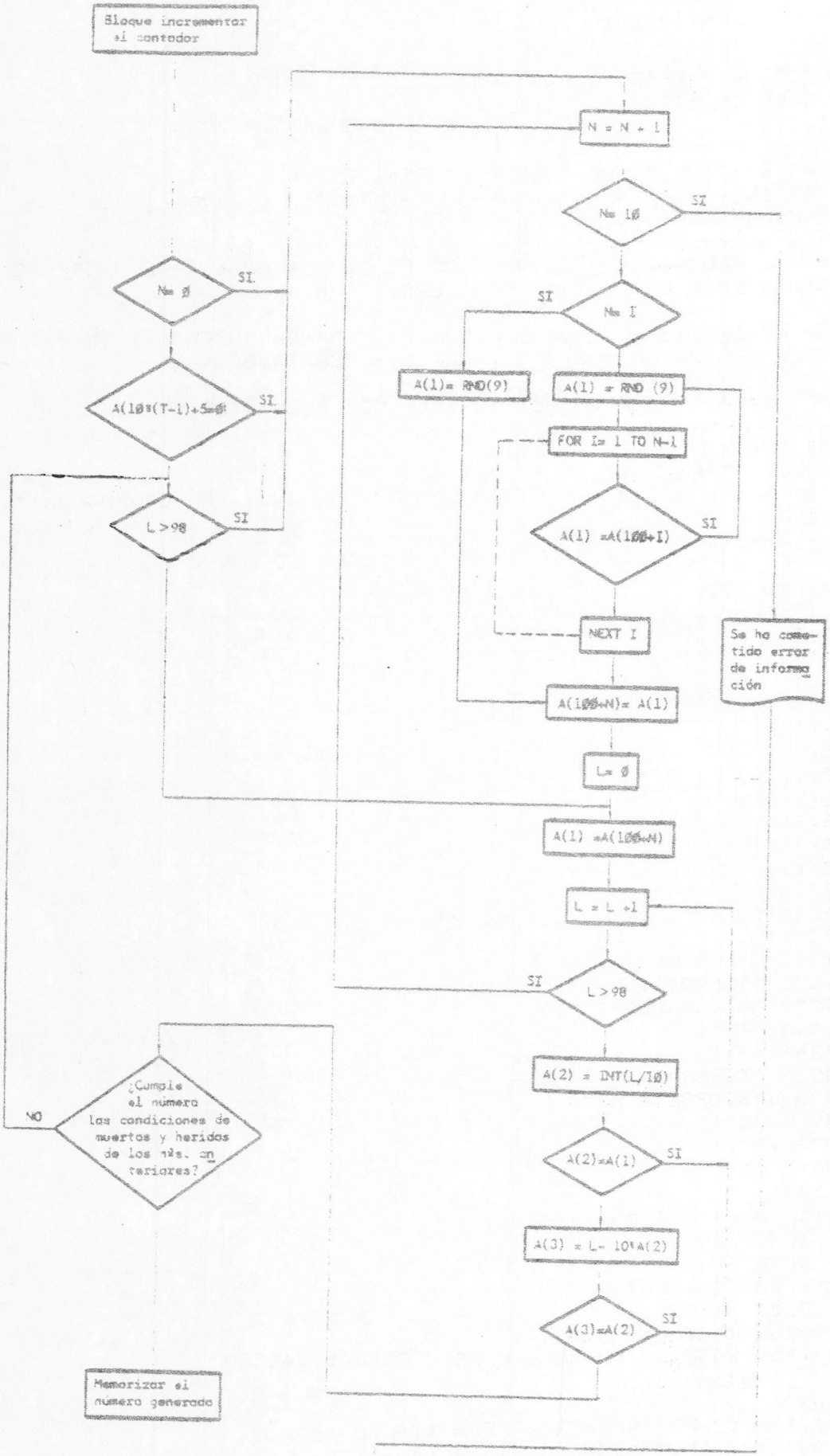
Si se ha editado el programa correctamente y se ejecuta se puede apreciar en términos reales su lentitud de respuesta en las últimas tiradas del ordenador. El motivo ha sido explicado en páginas anteriores.
Este es otro de los puntos importantes a tener en cuenta en la elaboración de un programa : "su rapidez".

En las páginas sucesivas se va a mostrar una variante de programación que imprime mayor velocidad al programa. Además va a ser posible detectar la posibilidad de que el jugador introduzca errores de información en el juego.

La idea consiste en cambiar el generador de números al azar, por otro sistema que no repita el número. Un contador cumpliría perfectamente este requisito; sin embargo, si se elige un contador lineal (del 1 al 999, por ejemplo), el jugador sabría que eligiendo el nº "987", al ordenador le costaría mucho acertarle puesto que antes tendría que probar con todos los anteriores, y desde luego, nunca existiría la posibilidad de que acertara en las primeras tiradas.

El contador que se va a elegir, tiene la primera cifra aleatoria.
Cuando este contador haya generado todos los números, si se da la circunstancia de que todavía no ha acertado el número, sucede, sin duda que se han introducido errores en la información de muertos y heridos por el jugador.

DIAGRAMA DE FLUJO DE LAS MODIFICACIONES



PROGRAMA

```
10 REM *JUEGO DE MUERTOS Y HERIDOS*
15 REM *FICHERO NUMER 2*
16 DIM A(110)
20 CLS
30 PRINT TAB(18);"JUEGO DE MUERTOS Y HERIDOS"
40 PRINT TAB(18);"==========================="
50 PRINT "REGLAS DEL JUEGO"
60 PRINT "----------------"
70 PRINT"A) EL ORDENADOR Y EL JUGADOR ELIGEN UN NUMERO DE 3 CIFRAS DIS-"
80 PRINT"TINTAS QUE NO EMPIECE POR CERO."
85 PRINT
90 PRINT"B) DE MANERA ALTERNATIVA SE VAN TIRANDO NUMEROS E INFORMANDO "
100 PRINT"DEL NUMERO DE MUERTOS Y HERIDOS ACERTADOS."
105 PRINT
110 PRINT"C) GANA EL QUE ANTES CONSIGUE LOS 3 MUERTOS."
120 PRINT
130 T=0:J=0:N=0
140 PRINT"QUIEN EMPIEZA?"
150 PRINT"SI SE PULSA UN 1 (EL JUGADOR) Y CON UN 2 (EL ORDENADOR)";
160 INPUT E
170 IF (E<>1)AND(E<>2) THEN 120
180 CLS
200 FOR X=0 TO 127
210 SET(X,13)
220 SET(X,18)
230 SET(X,47)
240 NEXT X
250 FOR Y=13 TO 47
260 SET(0,Y)
270 SET(15,Y)
280 SET(43,Y)
290 SET(53,Y)
300 SET(63,Y)
310 SET(64,Y)
320 SET(79,Y)
330 SET(107,Y)
340 SET(117,Y)
350 SET(127,Y)
360 NEXT Y
370 PRINT@321,"TIRADA";
380 PRINT@329,"N.JUGADOR";
390 PRINT@343,"M.";
400 PRINT@348,"H.";
410 PRINT@353,"TIRADA";
420 PRINT@361,"N.ORDENADOR";
430 PRINT@375,"M.";
440 PRINT@380,"H.";
450 PRINT@0," ";
460 GOSUB 2000
500 A(5)=A(1):A(6)=A(2):A(7)=A(3)
510 IF E=1 GOTO 550
520 IF E=2 GOTO 800
550 PRINT@0,"EMPIEZA USTED                                    "
560 FORI=1TO500:NEXTI
570 J=J+1:PRINT@(386+J*64),J;
580 PRINT@0,"INTRODUZCA UN NUMERO DE 3 CIFRAS VALIDO"
590 FORI=1TO1000:NEXTI
600 PRINT@0,"                                                 "
610 PRINT@0," ";:INPUT"PRIMERA CIFRA";V
620 PRINT@0," ";:INPUT"SEGUNDA CIFRA";X
```

```
630 PRINT@0,"  ";:INPUT"TERCERA CIFRA";Y
640 PRINT@0,"                                        "
650 IF(V>9)OR(X>9)OR(Y>9) THEN 690
660 IF((V-INT(V))<>0)OR((X-INT(X))<>0)OR((Y-INT(Y))<>0) THEN 690
670 IF(V<1)OR(V=X)OR(V=Y)OR(X=Y)OR(X<0)OR(Y<0) THEN 690
680 GOTO 710
690 PRINT@0,"EL NUMERO NO ES VALIDO. ESCRIVALE DE NUEVO"
700 GOTO 590
710 PRINT@(394+J*64),V;X;Y;
720 A(1)=A(5):A(2)=A(6):A(3)=A(7)
730 GOSUB 3000
740 PRINT@(407+J*64),M;
750 PRINT@(412+J*64),H;
760 IF M=3 GOTO 1200
780 PRINT@0,"NO HA ACERTADO Y AHORA ME TOCA A MI"
790 GOTO 820
800 PRINT@0,"                                        "
810 PRINT@0,"EMPIEZO YO"
820 FORI=1TO750:NEXTI
830 PRINT@0,"                                        "
840 PRINT@0,"ESPERE A QUE CALCULE EL NUMERO"
850 FORI=1TO500:NEXTI
860 T=T+1
870 PRINT@(418+T*64),T;
880 IF N=0 GOTO 6000
890 IF A(10*(T-1)+5)=0 GOTO 6000
900 IF L>98 GOTO 6000
910 GOTO 6110
920 IF T=1 GOTO 990
930 FOR K=1 TO T-1
940 V=A(10*K+1):X=A(10*K+2):Y=A(10*K+3)
950 GOSUB 3000
960 IF(M=A(10*K+5))AND(H=A(10*K+6)) GOTO 980
970 GOTO 900
980 NEXT K
990 A(10*T+1)=A(1):A(10*T+2)=A(2):A(10*T+3)=A(3)
1000 PRINT@0,"MI NUMERO ES EL :";A(1);A(2);A(3);"          "
1010 FORI=1TO750:NEXTI
1020 PRINT@(426+64*T),A(1);A(2);A(3);
1030 PRINT@0,"INTRODUZCA MUERTOS Y HERIDOS";
1040 FORI=1TO750:NEXTI
1050 PRINT@0,"                                        "
1060 PRINT@0,"  ";:INPUT"NUMERO DE MUERTOS";M
1070 PRINT@0,"                                        "
1080 PRINT@0,"  ";:INPUT"NUMERO DE HERIDOS";H
1090 IF(M>3)OR(H>3)OR(M+H>3)OR(M-INT(M)<>0)OR(H-INT(H)<>0) GOTO 1030
1100 IF(M<0)OR(H<0)OR(M=2)AND(H=1) GOTO 1030
1110 PRINT@0,"                                        "
1120 PRINT@(439+64*T),M;
1130 PRINT@(444+64*T),H;
1140 A(10*T+5)=M:A(10*T+6)=H
1150 IF M=3 THEN 1180
1160 PRINT@0,"ES SU TURNO";
1170 GOTO 560
1180 PRINT@0,"HE GANADO. MI NUMERO ERA EL";A(5);A(6);A(7)
1190 GOTO 1999
1200 PRINT@0,"ENHORBUENA, HA GANADO USTED"
1999 GOTO 1999
2000 FOR I=1 TO 3
2010 A(I)=RND(10)-1
2020 NEXT I
2030 IF(A(1)=0)OR(A(1)=A(2))OR(A(1)=A(3))OR(A(2)=A(3)) THEN 2000
2040 RETURN
3000 M=0:H=0
3010 IF V=A(1) GOSUB 4000
3020 IF X=A(1) GOSUB 5000
3030 IF Y=A(1) GOSUB 5000
3040 IF V=A(2) GOSUB 5000
```

```
3050 IF X=A(2) GOSUB 4000
3060 IF Y=A(2) GOSUB 5000
3070 IF V=A(3) GOSUB 5000
3080 IF X=A(3) GOSUB 5000
3090 IF Y=A(3) GOSUB 4000
3100 RETURN
4000 M=M+1
4010 RETURN
5000 H=H+1
5010 RETURN
6000 N=N+1
6010 IF N=10 GOTO 6200
6020 IF N=1 GOTO 6080
6030 A(1)=RND(9)
6040 FOR I=1 TO N-1
6050 IF A(1)=A(100+I) GOTO 6030
6060 NEXT I
6070 GOTO 6090
6080 A(1)=RND(9)
6090 A(100+N)=A(1)
6100 L=0
6110 A(1)=A(100+N)
6120 L=L+1
6130 IF L>98 GOTO 6000
6140 A(2)=INT(L/10)
6150 IF A(2)=A(1) GOTO 6120
6160 A(3)=L-10*A(2)
6170 IF A(3)=A(1) GOTO 6120
6180 IF A(3)=A(2) GOTO 6120
6190 GOTO 920
6200 PRINT@0,"REVISE MUERTOS Y HERIDOS.SE HA COMETIDO ERROR ."
6210 PRINT"EL JUEGO NO ES VALIDO,HAY QUE COMENZAR UNO NUEVO."
6220 GOTO 1999
```

INTRODUCCION

El título de este apartado refleja en buena medida el objetivo perseguido. Evidentemente, el campo científico-técnico es tan amplio y sofisticado que cada día se requiere un grado de especialización mayor; de cualquier forma, resulta imprescindible para acceder a él, el dominio de ciertas materias básicas tales como matemáticas, física, etc.... Estas materias, constituyen sin lugar a dudas la piedra angular de cualquier desarrollo técnico o de investigación y su estudio se ha generalizado a casi todos los tipos y niveles de enseñanza.

Determinados temas de estas áreas, a menudo resultan difíciles de asimilar por el alumno debido a su carácter abstracto. Aquí es donde pensamos que el ordenador puede ofrecer alternativas de ayuda importantes.

Una de las principales características del ordenador reside en su rapidez de cálculo. Además permite visualizar procesos de manera secuencial. Estas dos cualidades constituyen la base de este apartado, en el estudio de procesos de cálculo.

Entendemos que existe una infinidad de temas o casos que podrían ser objeto de estudio y aplicación en este apartado; por ello se van a tratar sólo algunos atendiendo a un criterio de importancia, todo ello sin olvidar el nivel de enseñanza al que va dirigido.

El apartado puede constituir una fase intermedia entre el aprendizaje del lenguaje BASIC I y la introducción al BASIC II y otras aplicaciones concretas.

CALCULO DE LIMITES

Valiéndonos de nuestro microordenador, podemos adquirir el concepto de límite de una sucesión, sin gran dificualtad.

Una sucesión es un conjunto de números, que puede responder a una ley matemática.

Por ejemplo, el conjunto de números

$$\frac{1}{2}, \frac{3}{4}, \frac{7}{8}, \frac{15}{16}, \ldots\ldots$$

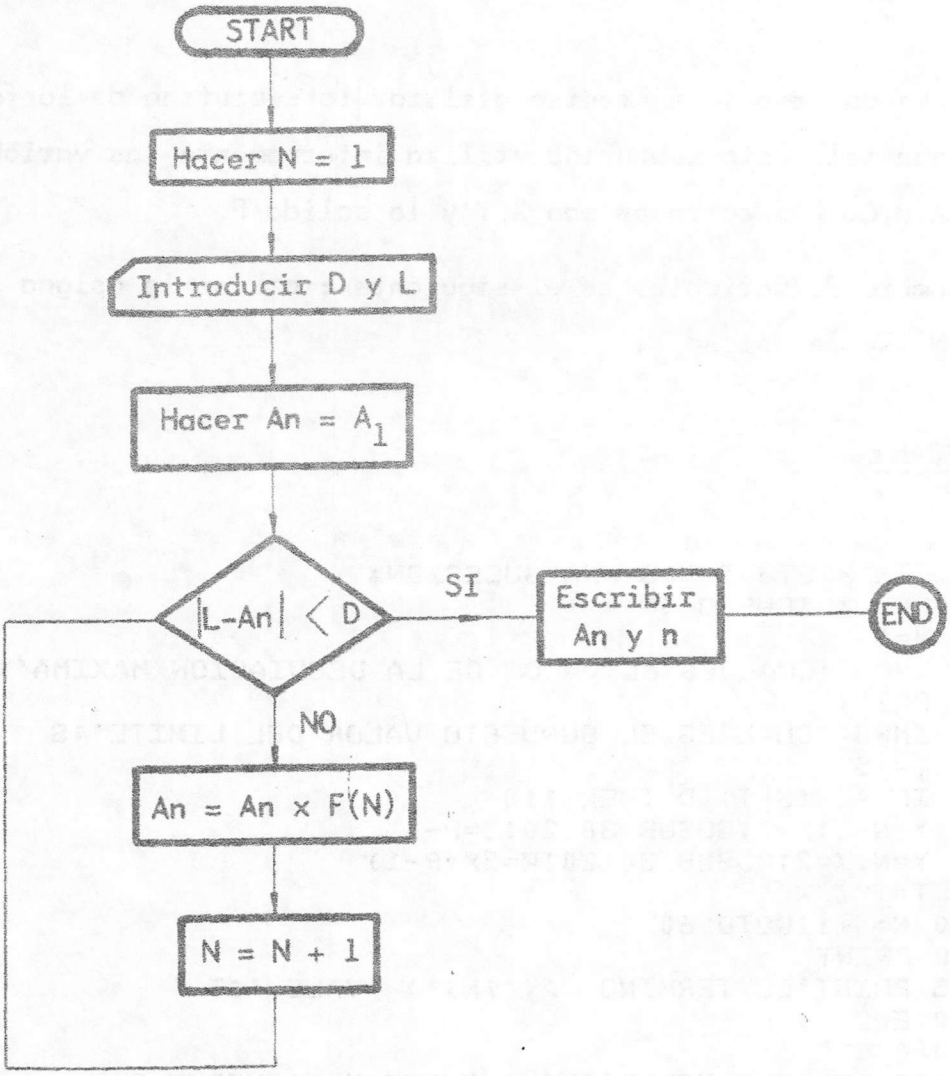
se puede obtener haciendo $An = \dfrac{2^N - 1}{2^N}$ y dándo valores sucesivos a n, tales como 1, 2, 3, 4 , ...

Si seguimos dando valores a n, los términos que obtenemos se van acercando cada vez más a un número fijo, que en este caso parece ser la unidad.

Diremos que el límite de la sucesión anterior es 1, si podemos hacer que la diferencia entre 1 y un término suficientemente avanzado de la sucesión sea en valor absoluto menor que cualquier cantidad por pequeña que sea.

Hagamos un programa que pueda servirnos para comprobar si una sucesión cualquiera An tiene un límite determinado L. Para ello supongamos conocida la relación $\dfrac{An + 1}{An} = F(N)$

Llamemos A_1 al primer término de la sucesión y D a la desviación máxima que deseamos tenga el término N-símo de nuestra sucesión respecto del límite L. El diagrama de flujo correspondiente a nuestro programa será el siguiente :



Examinando con atención el diagrama observaremos que si el límite escogido L es el verdadero, podemos acercar a él tanto como queramos el término N-simo de nuestra sucesión. Basta para ello con tomar un término suficientemente avanzado de la misma y esto lo consigue automáticamente el bucle que permite el incremento sucesivo de N.

En el ejemplo a que hacíamos referencia al comienzo de este apartado, la relación entre A_{N+1} y A_N es la siguiente :

$$F(N) = \frac{A_{N+1}}{A_N} = \frac{\dfrac{2^{N+1} - 1}{2^{N+1}}}{\dfrac{2^N - 1}{2^N}} = \frac{2^{N+1} - 1}{2(2^N - 1)}$$

Como se aprecia será preciso utilizar la subrutina de la función exponencial. Esta subrutina utiliza internamente las variables E,L,A,B,C. Las entradas son X,Y y la salida P.

El cambio de variables es el siguiente : a L se le asigna S, a A(N) se le asigna T.

PROGRAMA

```
10 REM *ESTUDIO DE UNA SUCESION*
15 REM *FICHERO (9)*
20 N=1
30 INPUT"CUAL ES EL VALOR DE LA DESVIACION MAXIMA";D
35 PRINT
40 INPUT"CUAL ES EL SUPUESTO VALOR DEL LIMITE";S
50 T=.5
60 IF ABS(S+T)<D THEN 110
70 Y=N+1:X=2:GOSUB 30120:Z=P-1
80 Y=N:X=2:GOSUB 30120:W=2*(P-1)
90 T=T*Z/W
100 N=N+1:GOTO 60
110 PRINT
115 PRINT"EL TERMINO  A(";N;")  VALE ";T
120 END
30000 END
30100 REM*EXPONENCIACION* INPUT X,Y;OUTPUT P
30110 REM*TAMBIEN USA INTERNAMENTE E,L,A,B,C*
30120 IF X=0 THEN P=0:RETURN
30130 P=1:IF Y=0 THEN RETURN
30140 IF(X<0)AND(Y-INT(Y)=0) THEN P=1-2*Y+4*INT(Y/2):X=-X
30150 GOSUB 30190:X=Y*L:GOSUB 30250
30160 P=P*E:RETURN
30170 REM*NATURAL & COMUN *INPUT X,OUTPUTL,X
30175 REM SALIDA L ES LOG NATURAL, SALIDA X ES LOG COMUN
30180 REM TAMBIEN UTILIZA A,B,C INTERNAMENTE
30190 E=0:IF X<0 THEN PRINT"LOG INDEFINIDO EN";X;STOP
30195 A=1:B=2:C=.5
30200 IF X>=A THEN X=C*X:E=E+A:GOTO 30200
30205 IF X<C THEN X=B*X:E=E-A:GOTO 30205
30210 X=(X-.707107)/(X+.707107):L=X*X
30215 L=(((.598979*L+.961471)*L+2.88539)*X+E-.5)*.693147
30220 IF ABS(L)<1E-6 THEN L=0
30225 X=L*.4342945:RETURN
30240 REM*EXPONENCIAL* INPUT X,OUTPUT E
30245 REM TAMBIEN USA INTERNAMENTE A,B *
30250 IF X=0 THEN E=1:RETURN
30255 IF INT(1.4427*X)>126 THEN PRINT "SOBRERRANGO":STOP
30260 IF X<0 GOTO 30280
30265 E=1:A=1:B=0
30270 IF A<.00001 THEN RETURN
30275 A=A*X/(B+1):E=E+A:B=B+1:GOTO 30270
30280 X=-X:GOSUB 30265:E=1/E:RETURN
```

CALCULO DE FUNCIONES "SUMA DE SERIES"

En matemáticas, se estudia la fórmula de Taylor como método de cálculo de funciones. Un ejemplo de ello, puede constituir el cálculo de la función exponencial e^x cuya serie desarrollada en serie de Taylor viene dada por la expresión :

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots\cdots + \frac{x^n}{n!} + Rn\ (x)$$

Rn es lo que se denomina "término complementario" y su valor tiende a cero para valores grandes de n

$$\lim_{n \to \infty} Rn(x) \to 0$$

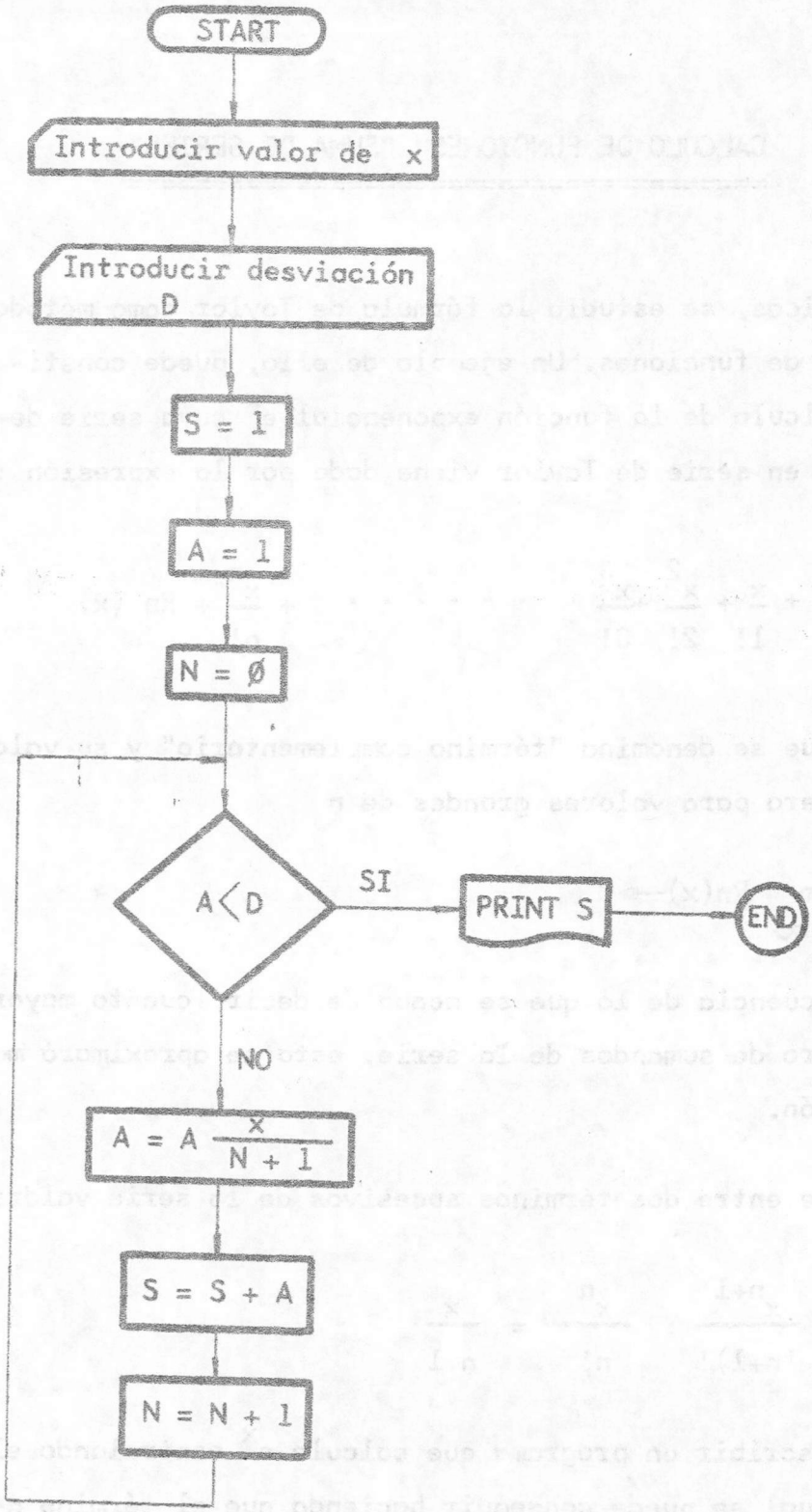
Como consecuencia de lo que se acaba de decir, cuanto mayor es el número de sumandos de la serie, esta se aproximará más a la función.

El cociente entre dos términos sucesivos de la serie valdría :

$$\frac{x^{n+1}}{(n+1)!} : \frac{x^n}{n!} = \frac{x}{n+1}$$

Se puede escribir un programa que calcule e^x controlando el error,lo cual se puede conseguir haciendo que el término n-simo sea inferior a un cierto valor prefijado.

El diagrama de flujo será el siguiente :



PROGRAMA

```
10 REM * SUMA DE SERIE * EXPONENCIAL
20 INPUT"CUAL ES EL VALOR DE X";X
30 INPUT"CUAL ES LA DESVIACION";D
40 S=1:A=1:N=0
50 IF A<D PRINT"EL VALOR DE LA SERIE ES:"; S:STOP
60 A=A*X/(N+1):S=S+A:N=N+1:GOTO 50
70 END
```

PREGUNTAS

¿Cuál es el valor del número e?

¿Para qué valor de D se obtienen 6 cifras exactas de este número?

EJERCICIOS PROPUESTOS :

1- Crear un programa que calcule la función trigonométrica seno x y otro para cos x. Las series son :

$$\text{sen } x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots\ldots\ldots + Rn\ (x)$$

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \ldots\ldots\ldots\ldots + Rn(x)$$

RAIZ DE UNA ECUACION

Con la ayuda del microcomputador, puede resultar sencillo el cálculo de la raíz de una ecuación.

La ecuación :

$$x + \log(x) = 0$$

tiene una raíz en el intervalo $0 < x < 1$. Se puede escribir un programa en BASIC que encuentre esta raíz y la escriba con un número determinado de cifras decimales exactas.

El método a seguir es el siguiente :

Para valores pequeños de x, F(x) = log (x) cuyo valor absoluto es grande y su signo negativo. Para x = 1, F(x) = 1.

Según esto, debe haber algún valor de x entre x = 0 y x = 1 para el que F(x) tiene que ser cero.

Si se hace A = 1 y B = Ø.ØØØ1 y se escoge un valor de x en la mitad de ese intervalo, ($\bar{x}$ = (A + B)/2) y se pueden sacar las siguientes conclusiones :

1º) Si para ese valor de $\bar{x}$, F(x) tiene el mismo signo que F(A), la raíz de la ecuación está entre B y $\bar{x}$, fig a).

2º) Si, en cambio, para ese valor de $\bar{x}$, F($\bar{x}$) tiene el mismo signo que F(B), la raíz tiene que estar entre $\bar{x}$ y A, fig b).

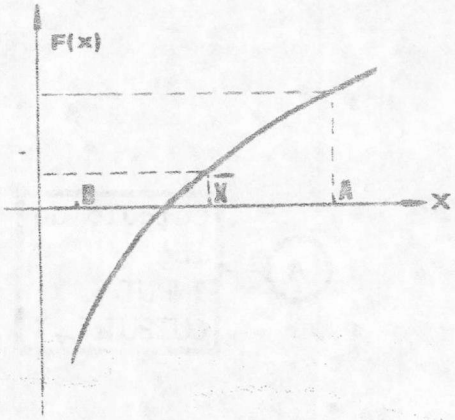


Fig. a)

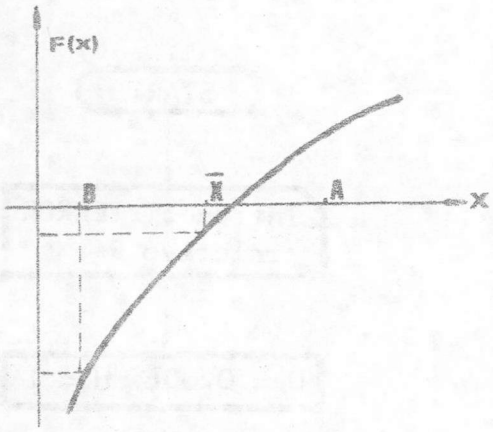


Fig. b)

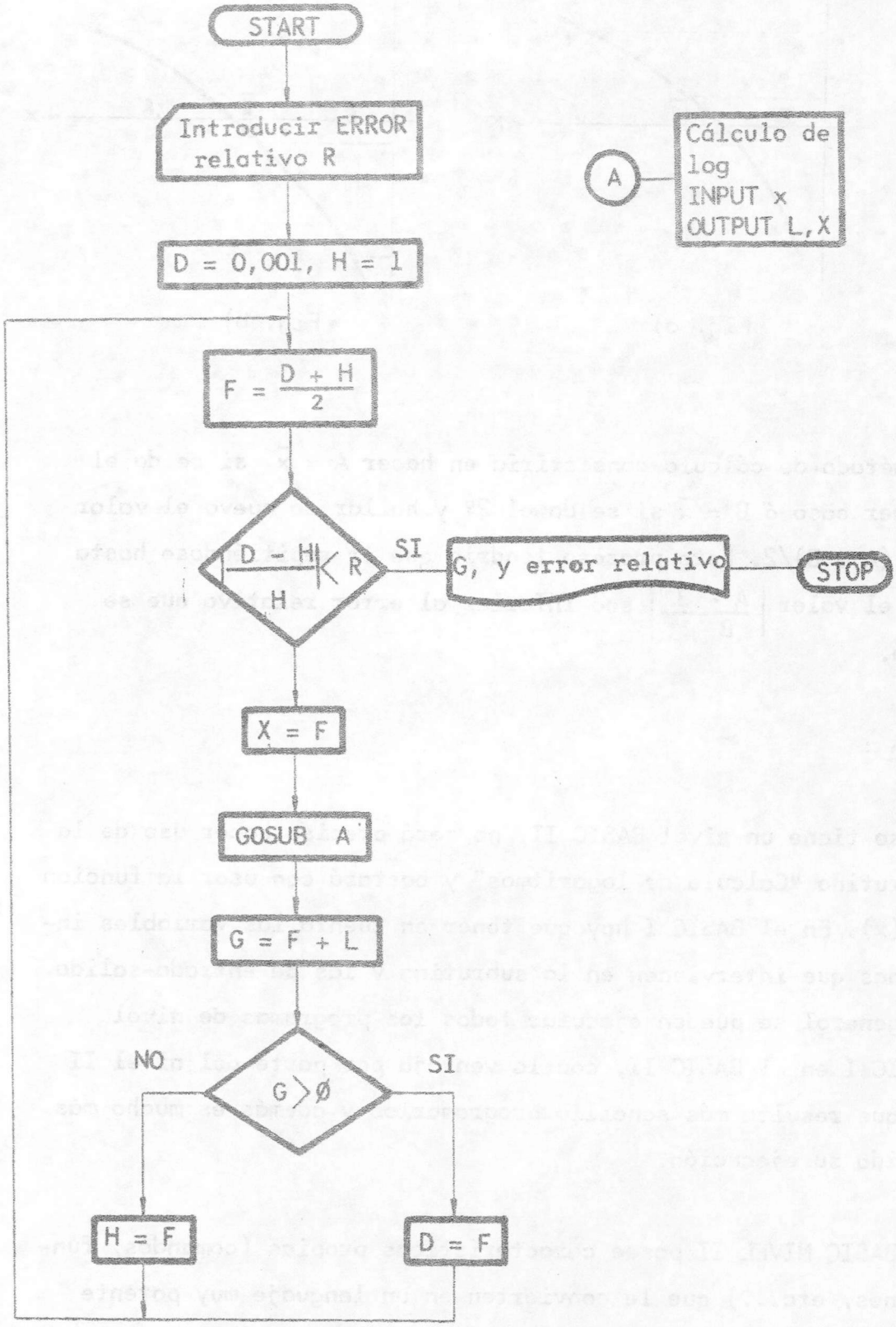
El método de cálculo consistiría en hacer $A = \bar{x}$ si se da el primer caso ó $B = \bar{x}$ si se da el 2º y hallar de nuevo el valor $\bar{x} = (A + B)/2$. Este proceso tendría que ir repitiéndose hasta que el valor $\left| \frac{A - B}{B} \right|$ sea inferior al error relativo que se fije.

NOTA :

Si se tiene un nivel BASIC II, no será preciso hacer uso de la subrutina "Cálculo de logaritmos" y bastará con usar la función ABS(x). En el BASIC I hay que tener en cuenta las variables internas que intervienen en la subrutina y las de entrada-salida. En general se pueden ejecutar todos los programas de nivel BASIC I en el BASIC II, con la ventaja por parte del nivel II de que resulta más sencillo programarlos y además es mucho más rápida su ejecución.

El BASIC NIVEL II posee características propias (comandos, funciones, etc...) que le convierten en un lenguaje muy potente en relación al nivel BASIC I.

A continuación se ofrece el diagrama de flujo :


START
Introducir ERROR relativo R
D = 0,001, H = 1
F = (D + H)/2
|(D - H)/H| < R
SI
G, y error relativo
STOP
X = F
GOSUB A
G = F + L
G > Ø
NO
SI
H = F
D = F
A
Cálculo de log
INPUT x
OUTPUT L,X

PROGRAMA

```
10 REM*CALCULO DE LA RAIZ DE UNA ECUACION*
20 REM*ECUACION F(X)=X+LOG(X)*
30 INPUT"CUAL ES EL VALOR DEL ERROR RELATIVO";R
40 D=.001:H=1
50 F=(D+H)/2
60 IF ABS((D-H)/H)<R GOTO 100
70 X=F:GOSUB 30190:G=F+L
80 IF G<0 THEN D=F:GOTO 50
90 H=F:GOTO 50
100 PRINT"LA RAIZ VALE";F
110 PRINT"EL ERROR RELATIVO ES";ABS((D-H)/H)
30000 END
30170 REM*NATURAL & COMUN *INPUT X,OUTPUTL,X
30175 REM SALIDA L ES LOG NATURAL, SALIDA X ES LOG COMUN
30180 REM TAMBIEN UTILIZA A,B,C,E INTERNAMENTE
30190 E=0:IF X<0 THEN PRINT"LOG INDEFINIDO EN";X;STOP
30195 A=1:B=2:C=.5
30200 IF X>=A THEN X=C*X:E=E+A:GOTO 30200
30205 IF X<C THEN X=B*X:E=E-A:GOTO 30205
30210 X=(X-.707107)/(X+.707107):L=X*X
30215 L=(((.598979*L+.961471)*L+2.88539)*X+E-.5)*.693147
30220 IF ABS(L)<1E-6 THEN L=0
30225 X=L*.4342945:RETURN
```

EJERCICIO PROPUESTO

1 - Elaborar un programa que calcule las raíces reales del polinomio :

$$x^4 - \frac{3}{20}x^3 + \frac{11}{20}x^2 - \frac{3}{20}x - \frac{9}{20} = 0$$

teniendo en cuenta que una de ellas se encuentra en el intervalo $(0 \rightarrow 1)$ y la otra en el $(0 \rightarrow -1)$.

CONCEPTO DE DERIVADA

VELOCIDAD

Aunque la derivada es un concepto matemático es más práctico y pedagógico su estudio desde la Física.

Todos tenemos idea de lo que es la velocidad. Decimos que el velocímetro de un automóvil está graduado en Km/hora y nos resulta sencillo el concepto de velocidad media, que obtenemos dividiendo el espacio recorrido entre el tiempo empleado.

El problema surge cuando intentamos encontrar el significado de algo tan sencillo como : "Si en este momento volamos a 900 Km/hora".

Si decimos "en este momento", quiere decir que no se trata de una velocidad media, sino de una velocidad instantánea. ¿Y que es velocidad instantánea?. La respuesta a esta pregunta surge en forma de límite.

En efecto, supongamos que a partir de un cierto instante to transcurre un tiempo Δt durante el cual, el móvil considerado recorre un espacio ΔS. La velocidad media durante el intervalo de tiempo considerado es :

$$Vm = \frac{\Delta S}{\Delta t}$$

Para hallar la velocidad en el instante to, debemos hacer Δt lo más pequeño posible, razón por la cual podremos definir la velocidad instantánea para t = to como

$$V = \lim_{\Delta t \to 0} \frac{\Delta S}{\Delta t}$$

La definición a que hemos llegado sin pretenderlo, es la derivada de la función S respecto de t para t = to y se expresa así :

$$V = \frac{dS}{dt}_{t=to}$$

Consideremos un móvil lanzado con velocidad inicial Vo = 8 m/seg. y sometido a una aceleración a = 10 m/s^2. El espacio recorrido en el tiempo t, vale

$$S = 8 \cdot t + \frac{1}{2} 10\, t^2 = 8t + 5t^2$$

Supongamos que sólo disponemos de esta función y que deseamos hallar la velocidad para distintos valores de t. Comencemos por t= 1 para el que S = 13. Si a partir de t = 1 transcurre una décima de segundo, el nuevo valor de t será t = 1,1 en cuyo caso S = 8 . 1,1 + 5 . 1,1^2 = 14,85 ; Así pues :

$$\Delta t = 1'1 - 1 = 0,1$$

$$\Delta S = 14,85 - 13 = 1,85$$

$$V = \frac{\Delta S}{\Delta t} = \frac{1,85}{0,1} = 18,5$$

Se comprende fácilmente, que dando valores más pequeños a Δt, nos acerquemos más al verdadero valor de V. Veámoslo con el siguiente programa :

PROGRAMA

```
10 REM * CONCEPTO DE DERIVADA *
20 PRINT"SE DESEA CALCULAR LAS VELOCIDADES INSTANTANEAS EN UN INTERVALO?"
30 PRINT"RESPONDER CON UN SI O CON UN NO";
40 INPUT A$
50 IF (A$<>"SI")AND(A$<>"NO") GOTO 20
60 PRINT
70 IF A$="SI" THEN 160
80 PRINT"SE VA A PROCEDER ENTONCES A CALCULAR LA VELOCIDAD EN UN INS-"
90 PRINT"TANTE T?"
100 PRINT
110 INPUT"CUAL ES EL VALOR DEL INSTANTE DESEADO";A
120 INPUT"CUAL ES EL INCREMENTO DE TIEMPO";C
130 D=A:GOSUB 300
140 PRINT"VELOCIDAD M/S.","TIEMPO S.","INCREMENTO DE TIEMPO S."
150 PRINT S/C,A,C :GOTO 280
160 PRINT"EL INTERVALO COMIENZA EN EL INSTANTE A-SEGUNDOS Y TERMINA EN"
170 PRINT"EL INSTANTE B-SEGUNDOS"
180 PRINT
190 INPUT"CUAL ES EL VALOR DE A (EN SEGUNDOS)";A
200 INPUT"CUAL ES EL VALOR DE B (EN SEGUNDOS)";B
205 INPUT "CUAL ES EL VALOR DEL INCREMENTO DE TIEMPO EN SEGUNDOS";C
210 CLS
220 PRINT"VELOCIDAD M/S.","TIEMPO S.","INCREMENTO DE TIEMPO S."
225 D=A
230 FOR I=A TO B STEP (B-A)/10
240 GOSUB 300
250 PRINT S/C,D,C
260 D=D+(B-A)/10
270 NEXT I
280 END
300 X=D+C
310 S=(8*X+5*X*X)-(8*D+5*D*D)
320 RETURN
```

Obtener mediante el programa anterior, los valores de V para diferentes valores de t. Observar que al dar menores incrementos a t (Δt), los resultados se acercan más al valor exacto. Los valores exactos corresponden a la función $V = 8 + 10t$.

NOTA :

No se deben escoger valores de Δt demasiado pequeños porque el error que comete el ordenador al efectuar las operaciones puede ser significativo y alterar el resultado de manera importante.

ACELERACION

En el programa anterior, cambiamos la función S(t) por la $V(t) = 8 + 10\ t$, lo que equivale a cambiar la línea 31Ø

```
31Ø  S = (8 + 1Ø*X) - (8 + 1Ø*D)
```

Procediendo del mismo modo que antes, obtendremos valores correspondientes a dV/dt en lugar de dS/dt.

¿Pero qué representa dV/dt?. A partir de la definición que hemos visto de derivada, podremos escribir.

$$\frac{dV}{dt} = \lim_{\Delta t \to 0} \frac{\Delta V}{\Delta t}$$

Pero $\frac{\Delta V}{\Delta t}$ es la aceleración media experimentada por un móvil en un tiempo Δt. De esta forma :

$$\frac{dV}{dt} = \lim_{\Delta t \to 0} \frac{\Delta V}{\Delta t} = \text{aceleración media}$$

Ejecutando el programa de derivación para la función $V = 8 + 10t$ y dándo a t distintos valores, llegamos a una conclusión lógica. La aceleración obtenida no es sino la elegida al comienzo del ejercicio para formar la función $S(t)$.

NOTA :

Hay que cambiar las líneas 22Ø y 14Ø. Donde pone VELOCIDAD M/S, se debe escribir : ACELERACION.

MOVIMIENTO ARMONICO SIMPLE

Nada mejor para completar el concepto de derivada tratado en el ejercicio anterior, que mediante el estudio de un fenómeno físico como el del enunciado. Existen gran cantidad de ejemplos físicos de movimiento armónico simple entre los que se encuentran : El movimiento de una masa atada a un muelle, el ejemplo del péndulo, etc... Aquí nos vamos a fijar sin embargo, en uno muy sencillo que simplifica los cálculos.

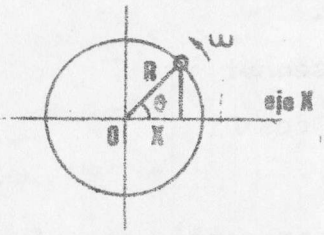


Fig. a)

La figura a) muestra una pelota atada por una cuerda (R) que gira a velocidad angular constante ω sobre el punto "O".
El desplazamiento de la sombra de la pelota sobre el eje X constituye un claro ejemplo de movimiento armónico simple.

La abcisa x recibe el nombre de elongación y tendrá por valor :

$$X = R \cos \theta$$

El ángulo girado por el radio vector que une el centro del círculo con la pelota crece uniformemente, por lo que :

$$\theta = \omega t$$

Sustituyendo este valor en la expresión anterior obtenemos :

$$X = R \cos \omega t$$

Hemos establecido una relación entre el desplazamiento X de la sombra y el tiempo t. Así pues, $X = f(t)$
Por otra parte ω es igual a $2\pi f$ donde f es la frecuencia (ciclos por segundo).

Recordando páginas anteriores ...

$$V = \lim_{\Delta t \to 0} \frac{\Delta X}{\Delta t} = \frac{d\,X}{d\,t}$$ ó si se prefiere utilizar

otra nomenclatura, $V = f'(t)$

De esta forma $a = V'(t)$ ó lo que es igual $f''(t)$. Como se sabe la derivada de (sen x) es igual a (cos x dx) y la de (cos x) es igual a (- sen x dx).

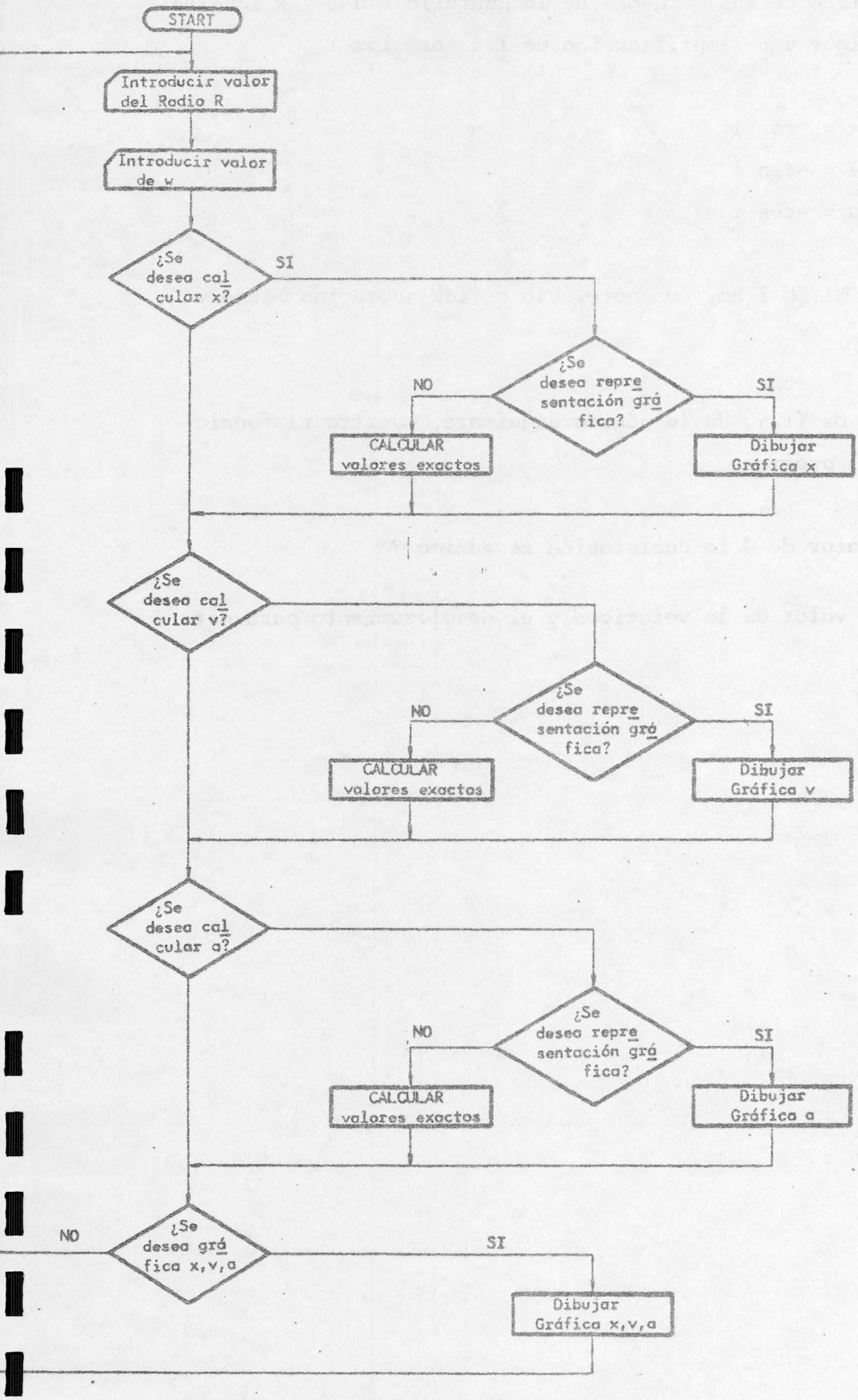
Resumiendo :

Desplazamiento $X = f(t) = R \cos \omega t$

Velocidad $V = f'(t) = - R\,\omega \operatorname{sen} \omega t$

Aceleración $A = f''(t) = - R\,\omega^2 \cos \omega t$

Se puede elaborar un programa que represente estas funciones en forma gráfica, de forma que dé entrada a valores de ω y Xo. Estos valores no deben sobrepasar la unidad.


START
Introducir valor del Radio R
Introducir valor de w
¿Se desea calcular x?
SI
¿Se desea representación gráfica?
NO
SI
CALCULAR valores exactos
Dibujar Gráfica x
¿Se desea calcular v?
¿Se desea representación gráfica?
NO
SI
CALCULAR valores exactos
Dibujar Gráfica v
¿Se desea calcular a?
¿Se desea representación gráfica?
NO
SI
CALCULAR valores exactos
Dibujar Gráfica a
¿Se desea gráfica x,v,a
NO
SI
Dibujar Gráfica x,v,a

Para no salirse de los margenes de la pantalla con ω y R iguales a 1 se consigue una simplificación de las fórmulas :

$$x = \cos t$$

$$v = -\text{sen } t$$

$$a = -\cos t$$

En el BASIC NIVEL I hay que hacer uso de las subrutinas seno y coseno.

El diagrama de flujo de la página siguiente, muestra el funcionamiento del programa.

¿Para qué valor de θ la aceleración es máxima ?

¿Cual es el valor de la velocidad y el desplazamiento para este ángulo ?

OGRAMA

```
   REM * ESTUDIO DEL MOVIMIENTO ARMONICO SIMPLE *
12 CLS
15 PRINT TAB(9);"-----------------------------------------"
   PRINT TAB(10);"ESTUDIO DEL MOVIMIENTO ARMONICO SIMPLE"
   PRINT TAB(9);"-----------------------------------------"
40 INPUT"CUAL ES EL VALOR DEL RADIO (R<=1)";R
50 IF R>1 GOTO 40
   INPUT"CUAL ES EL VALOR DE W (W<=1)";W
   IF W>1 GOTO 60
75 F=INT(10*((2*3.1416)/W))/10:M=0
80 INPUT"SE DESEA CALCULAR EL DESPLAZAMIENTO X";A$
   IF (A$<>"SI")AND(A$<>"NO") GOTO 80
   0 IF A$="SI" GOTO200
110 INPUT"SE DESEA CALCULAR LA VELOCIDAD V";A$
120 IF (A$<>"SI")AND(A$<>"NO") GOTO 110
   30 IFA$="SI" GOTO 500
   0 INPUT"SE DESEA CALCULAR LA ACELERACION A";A$
150 IF(A$<>"SI")AND(A$<>"NO")GOTO 140
160 IF A$="SI"GOTO 800
   70 PRINT"SE DESEA REPRESENTAR LAS 3 GRAFICAS (X,V,A)";
   30 INPUT A$
190 IF(A$<>"SI") GOTO 10
195 GOTO 1100
   00 PRINT"SI SE DESEA UNA REPRESENTACION GRAFICA DEL DESPLAZAMIENTO ,"
   10 PRINT"PULSAR UN (1); Y SI SE QUIEREN VALORES MAS EXACTOS ,UN (0) .";
220 INPUT U
230 IF(U<>1)AND(U<>0) PRINT"LEER DE NUEVO.":GOTO 200
   40 IF U<>1 GOTO 370
   50 CLS:GOSUB 2000
260 PRINT@0,"GRAFICA (DESPLAZAMIENTO) ";
270 PRINT" W=";W;"R./S.";" R=";R;"M.";" F=";F;"C./S.";
   75 Z=5
   80 FOR I=0 TO 360 STEP 3
290 X=I:GOSUB 30360:Y=-(Y*R*20)+24.5
300 SET(Z,Y):Z=Z+1
   10 NEXT I
   15 IF M=1 THEN RETURN
   20 GOSUB 2100
330 PRINT@960,"DESEA CONTINUAR";:INPUTA$
   40 IF(A$<>"SI") GOTO 330
   50 GOTO110
370 INPUT"CUAL ES EL VALOR DEL INCREMENTO DEL ANGULO";G
380 CLS:D=0
   90 PRINT"ANGULO (EN GRADOS)","DESPLAZAMIENTO"
   00 PRINT"==================","==============="
410 X=D:GOSUB 30350:Y=Y*R
420 IF D>360 GOTO 110
   30 PRINT D,,Y
   40 D=D+G:GOTO410
500 PRINT"SI SE DESEA UNA REPRESENTACION GRAFICA DE LA VELOCIDAD ,"
510 PRINT"PULSAR UN (1); Y SI SE QUIEREN VALORES MAS EXACTOS UN (0).";
   20 INPUT U
   30 IF(U<>1)AND(U<>0) PRINT"LEER DE NUEVO":GOTO 500
   40 IF U<>1 GOTO 670
550 CLS:GOSUB 2000
   60 PRINT@0,"GRAFICA (VELOCIDAD)";
   70 PRINT" W=";W;"R./S.";" R=";R;"M.";" F=";F;"C./S.";
580 Z=5
590 FOR I=0 TO 360 STEP 3
   00 X=I:GOSUB 30370:Y=Y*R*W*20+24.5
```

```
610 SET(Z,Y):Z=Z+1
620 NEXT I
625 IF M=1 THEN RETURN
630 GOSUB 2100
640 PRINT@960,"DESEA CONTINUAR";:INPUT A$
650 IF(A$<>"SI")GOTO 640
660 GOTO 140
670 INPUT"CUAL ES EL VALOR DEL INCREMENTO DEL ANGULO";G
680 CLS:D=0
690 PRINT"ANGULO (EN GRADOS)","DESPLAZAMIENTO"
700 PRINT"==================","==============="
710 X=D:GOSUB 30370:Y=-Y*R*W
720 IF D>360 GOTO 140
730 PRINT D,,Y
740 D=D+G:GOTO 710
800 PRINT"SI SE DESEA UNA REPRESENTACION GRAFICA DE LA ACELEACION,"
810 PRINT"PULSAR UN (1); Y SI SE QUIEREN VALORES MAS EXACTOS UN (0).";
820 INPUT U
830 IF(U<>1)AND(U<>0) PRINT"LEER DE NUEVO.":GOTO 800
840 IF U<>1 GOTO 970
850 CLS:GOSUB 2000
860 PRINT@0,"GRAFICA (ACELERACION)";
870 PRINT" W=";W;"R./S.";" R=";R;"M.";" F=";F;"C./S.";
880 Z=5
890 FOR I=0 TO 360 STEP 3
900 X=I:GOSUB 30360:Y=(Y*R*W*W*20)+24.5
910 SET(Z,Y):Z=Z+1
920 NEXT I
925 IF M=1 THEN RETURN
930 GOSUB 2100
940 PRINT@960,"DESEA CONTINUAR";:INPUT A$
950 IF(A$<>"SI") GOTO 940
960 GOTO 170
970 INPUT"CUAL ES EL VALOR DEL INCREMENTO DEL ANGULO";G
980 CLS:D=0
990 PRINT"ANGULO (EN GRADOS)","DESPLAZAMIENTO"
1000 PRINT"==================","==============="
1010 X=D:GOSUB 30350:Y=-Y*R*W*W
1020 IF D>360 GOTO 170
1030 PRINT D,,Y
1040 D=D+G:GOTO 1010
1100 CLS:GOSUB 2000
1110 PRINT@0,"(COS.=X)";"(-SEN=V)";"(-COS.=A)";" W=";W;" R=";R;" F=";F;
1120 M=1:GOSUB 275:GOSUB 580:GOSUB 880:GOSUB 2100
1130 PRINT@960,"DESEA VOLVER A EMPEZAR";:INPUT A$
1140 IF(A$<>"SI") GOTO 1130
1150 GOTO 10
2000 REM*SUBRUTINA DIBUJO DE EJES*
2010 FOR X=5 TO 126
2020 SET(X,24)
2030 NEXT X
2040 FOR Y=3 TO 45
2050 SET(5,Y)
2060 NEXT Y
2070 SET(4,4):SET(4,44):PRINT@65,"1";:PRINT@896,"-1";
2080 PRINT@513,"0";:RETURN
2100 PRINT@529,"90";:PRINT@543,"180";:PRINT@558,"270";:PRINT@573,"360";
2110 RETURN
30000 END
30350 REM *COSENO* INPUT X EN GRADOS ,OUTPUT Y
30360 X=X+270:GOSUB 30380:Y=-Y:RETURN
30370 REM *SENO* INPUT X EN GRADOS, OUTPUT Y
30380 IFX>360 THEN X=X/360:X=(X-INT(X))*360
30390 IFX>90THENX=X/90:Y=INT(X):X=(X-Y)*90:ONYGOTO30410,30420,30430
30400 X=X/57.29578:GOTO 30440
30410 X=90-X:GOTO 30400
30420 X=-X:GOTO 30400
30430 X=X-90:GOTO30400
30440 Y=X-X*X*X/6+X*X*X*X*X/120-X*X*X*X*X*X*X/5040
30450 Y=Y+X*X*X*X*X*X*X*X*X/362880
30455 RETURN
```

CONCEPTO DE INTEGRAL

Del mismo modo que para hablar de velocidad instantánea necesitamos acudir al concepto de derivada, así también para hablar de trabajo habremos de acudir al concepto de integral.

Sabemos que :

Trabajo = Fuerza x desplazamiento

(Consideramos por simplicidad que la fuerza se halla en la dirección del desplazamiento).

Si suponemos que la fuerza es constante, el cálculo del trabajo se reduce a un mero producto de dos factores. Pero, ¿qué ocurre si la fuerza varía a lo largo del desplazamiento?

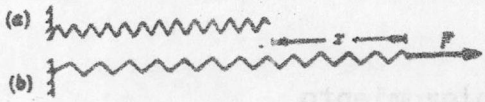
En el caso de que la fuerza varíe una solución aproximada la encontraremos parcelando el desplazamiento total en pequeños desplazamientos elementales. A lo largo de uno de estos pequeños desplazamientos podemos considerar que la fuerza se mantiene constante y el trabajo a lo largo del mismo será fácilmente calculable. La suma de todos los trabajos elementales cuando los respectivos desplazamientos se consideran infinitamente pequeños nos dará exactamente el valor del trabajo y dicha suma recibe el nombre de integral de la fuerza (variable) a lo largo del camino o desplazamiento considerado.

Se expresa así :

$$T = \int_a^b F \, dx$$

en donde a y b son las abcisas inicial y final del desplazamiento, respectivamente.

Mediante un ejemplo nos será fácil comprender la necesidad de este concepto matemático, que juntamente con el de derivada ha prestado un aporte fundamental para el desarrollo de la Ciencia y Técnica actuales.

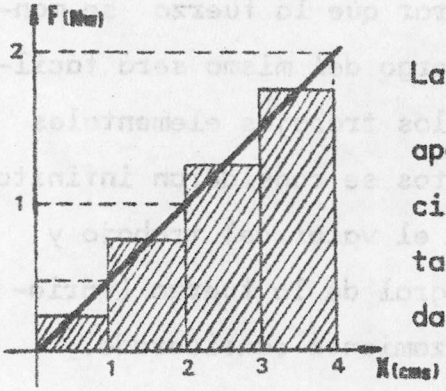


Consideremos un sencillo muelle reposando sobre una superficie horizontal y fijo por uno de sus extremos. Para desplazar el otro extremo un espacio X, hemos de aplicar una fuerza que en valor absoluto vale

$$F = kx$$

en donde k es una constante que depende de la elasticidad del muelle.

Supongamos que k valé 1/2 Nw/cm. y que queremos calcular el trabajo realizado para desplazar el muelle 4 cm. a partir de su posición de reposo.



La representación gráfica de $F = \frac{1}{2} x$ aparece en la figura y se comprueba fácilmente que se trata de una línea recta que pasa por el origen de coordenadas.

Para calentar el trabajo realizado al alargar el muelle 4 cms. podemos dividir el espacio en 4 partes iguales, averiguando el trabajo realizado en cada una de ellas.

A lo largo del primer centímetro de recorrido, la fuerza tampoco es constante, pero podemos considerar que lo es, adjudicándole el valor que posea en el punto medio del intervalo, es decir, para X = 0,5 cm.

Así pues, en el primer centímetro o primer desplazamiento elemental de la fuerza, el trabajo será :

$$\Delta T_1 = \Delta x . F = 1 . \frac{1}{2} . 0,5 = 0,25 \quad \text{ya que} \quad F = \frac{1}{2} . 0,5$$

Observaremos que ΔT, coincide con el área del primer rectángulo rayado.

Procederemos de forma análoga para calcular los demás trabajos elementales, así como el trabajo total. Los resultados vienen dados en el siguiente cuadro :

Nº del intervalo (N)	Fuerza en el centro del intervalo $F = \frac{1}{2} x$	Trabajo elemental $\Delta T = F . \Delta x$	Trabajo total T
1	$\frac{1}{2} 0,5 = 0,25$	$0,25 \times 1 = 0,25$	0,25
2	$\frac{1}{2} 1,5 = 0,75$	$0,75 \times 1 = 0,75$	0,25+0,75=1
3	$\frac{1}{2} 2,5 = 1,25$	$1,25 \times 1 = 1,25$	1+1,25=2,25
4	$\frac{1}{2} 3,5 = 1,75$	$1,75 \times 1 = 1,75$	2,25+1,75=4

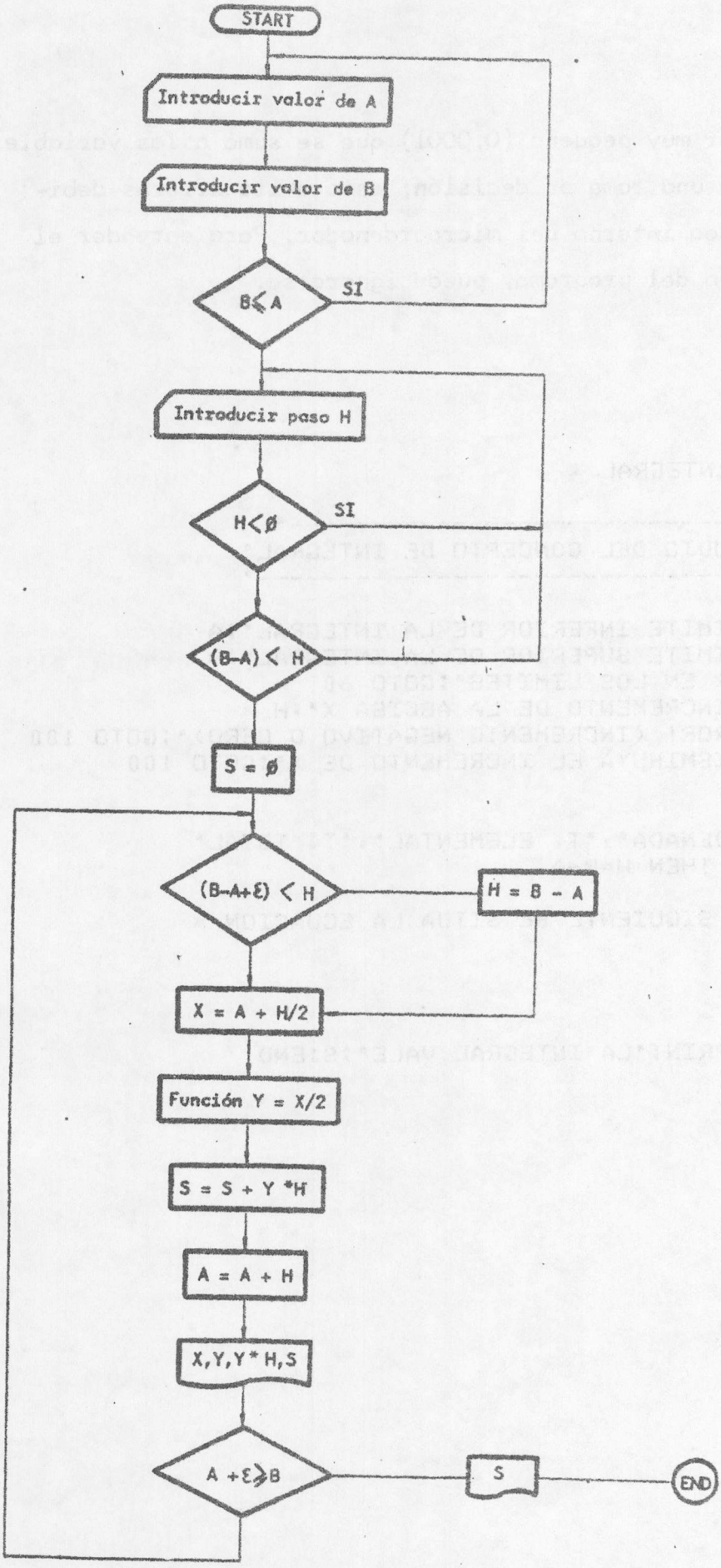
El trabajo total realizado al final de cada intervalo se obtiene sumando al trabajo acumulado anterior el trabajo elemental (ΔT) correspondiente a dicho intervalo. Así pues, el trabajo realizado en cada momento viene representado por la suma de las áreas de los rectángulos, cada uno de los cuales representa, a su vez, un trabajo elemental.

El error que cometemos en un caso general al considerar constante la fuerza a lo largo de un intervalo será tanto menor cuanto más pequeño sea el intervalo. En el caso límite de que pudiéramos hacer cada intervalo infinítamente pequeño no cometeríamos error alguno. Esto es precisamente lo que consigue hacer la integral. La suma de las áreas de los rectángulos será entonces el área comprendida entre la curva F(x), el eje de abcisas y las ordenadas extremas.

Así pues, la integral no sólo nos calcula un trabajo, sino también un área.

A continuación vamos a crear un programa que reproduzca la tabla de la página anterior. Como sabemos, la suma del trabajo total representa el valor de la integral definida.

A continuación aparece el diagrama de flujo :


START
Introducir valor de A
Introducir valor de B
B≤A
SI
Introducir paso H
H<Ø
SI
(B-A) < H
S = Ø
(B-A+ε) < H
H = B - A
X = A + H/2
Función Y = X/2
S = S + Y *H
A = A + H
X,Y,Y * H,S
A +ε≥B
S
END

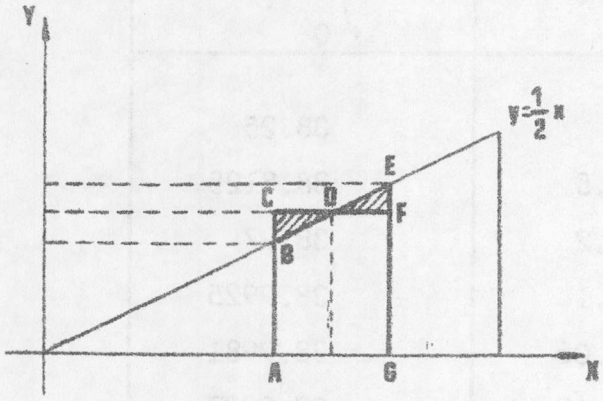
NOTA :

ε es un valor muy pequeño (0,0001) que se suma a las variables para efectuar una toma de decisión; esto evita errores debidos al redondeo interno del microordenador. Para entender el funcionamiento del programa, puede ignorarse.

PROGRAMA

```
10 REM * CONCEPTO DE INTEGRAL *
20 CLS
30 PRINT TAB(13);"--------------------------------"
40 PRINT TAB(13);"ESTUDIO DEL CONCEPTO DE INTEGRAL"
50 PRINT TAB(13);"--------------------------------"
60 PRINT
70 INPUT"CUAL ES EL LIMITE INFERIOR DE LA INTEGRAL";A
80 INPUT"CUAL ES EL LIMITE SUPERIOR DE LA INTEGRAL";B
90 IF B<=A PRINT"ERROR EN LOS LIMITES":GOTO 60
100 INPUT"CUAL ES EL INCREMENTO DE LA ABCISA X";H
110 IF H<=0 PRINT"!ERROR! (INCREMENTO NEGATIVO O CERO)":GOTO 100
120 IF(B-A)<H PRINT"DISMINUYA EL INCREMENTO DE X":GOTO 100
130 S=0
132 CLS
135 PRINT"ABCISA","ORDENADA","T. ELEMENTAL","T. TOTAL"
140 IF (B-A+.00001)<H THEN H=B-A
150 X=A+H/2
160 REM * EN LA LINEA SIGUIENTE SE SITUA LA ECUACION *
170 Y=3*X*X+2*X+1
180 S=S+Y*H
190 A=A+H
200 PRINT X,Y,Y*H,S
210 IF (A+.00001)>=B PRINT"LA INTEGRAL VALE";S:END
220 GOTO 140
```

Al ejecutar el programa, se puede observar que para un intervalo cualquiera (en nuestro ejemplo era de A = 0 hasta B = 4) el área, ó lo que es lo mismo la integral, da como resultado el mismo valor sea cual fuere el valor del incremento de la abcisa (Δx). Esto es debido a que la función a integrar es una recta en cuyo caso el cálculo de las áreas elementales es un cálculo exacto como muestra la figura.



TRIANGULO DBC = TRIANGULO DEF

AREA DEL RECTANGULO ACFG = AREA DEL TRAPECIO ABEG

Sin embargo, en un caso general, la función a integrar no tiene por qué ser una recta y por tanto el cálculo de las áreas elementales no se ajustarán al valor real. En estos casos, como podremos apreciar seguidamente, el cálculo de la integral definida será más pequeña cuanto menor sea elΔx elegido.

En efecto : supongamos que queremos hallar la integral definida de la parábola $y = 3x^2 + 2x + 1$ en el intervalo [A =0, B= 3] vamos a establecer un cálculo matemático previo, para poder efectuar comparaciones con los resultados que obtengamos al ejecutar nuestro programa.

$$\int_0^3 3x^2 + 2x + 1 = \left[x^3 + x^2 + x\right]_0^3 = \left[27 + 9 + 3\right] - \left[0\right] =$$

$$= \boxed{39}$$

Antes de ejecutar el programa, hay que cambiar la línea 17Ø

```
17Ø   y = 3 *X * X + 2 * X + 1
```

Si se dan distintos valores aΔx se obtienen los siguientes resultados :

ΔX	$\int_0^3 f(x)$
1	38.25
0.5	38.8125
0.2	38.97
0.1	38.9925
0.05	38.9981
0.02	38.9997
0.01	38.9999

Se puede así mismo apreciar la ejecución con intervalos negativos.

Ejecutar el programa para los valores A = -1 y B = + 1. El resultado debe aproximarse a 4.

APLICACIONES DE LA INTEGRAL DEFINIDA

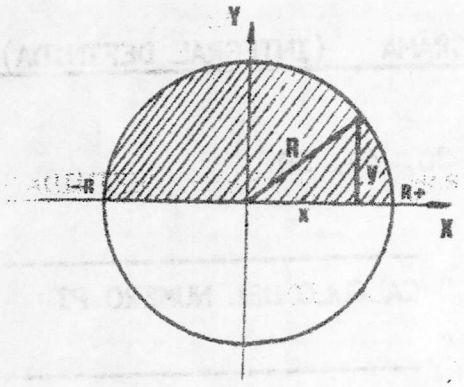
CALCULO DE π

Como se ha dicho, la integral definida puede asociarse al cálculo de un determinado área en el plano XY (llamado plano R^2).

De aquí que la aplicación más inmediata de la integral definida esté en el cálculo de áreas de figuras trigonométricas o de curvas.

Más adelante calcularemos alguna de estas áreas, pero ahora vamos a fijar nuestra atención en el cálculo del número π.

Vamos a demostrar con ayuda del microordenador, la existencia de dicho número, y luego vamos a calcular su valor, Para ello vamos a estudiar el área del círculo.



La ecuación de la circunferencia centrada en los ejes es :

$$x^2 + y^2 = R^2 \quad ; \quad y = \pm\sqrt{R^2 - x^2}$$

Vamos a crear un programa que calcule el área del semicírculo superior. Naturalmente, usaremos una integral definida para este propósito y los límites de integración serán en este caso $A = -R$ $B = +R$.

Se puede utilizar el programa del cálculo de la integral definida estudiado, efectuando las siguientes modificaciones :

1- Las entradas de datos del programa serán : el valor del radio y el paso de integración.

2- La ecuación $y = f(x)$ es en este caso $y = \sqrt{R^2 - x^2}$. Si se tiene un BASIC NIVEL I habrá que hacer uso de la subrutina raíz cuadrada. En caso de tener un BASIC NIVEL II se usará la función raíz cuadrada (SQR). La rapidez de ejecución del programa en este último caso es superior al primero.

3- Una vez calculada la integral definida S, vamos a mandar imprimir los valores :

Radio del círculo = R

Area del círculo = 2.S

La relación $= \dfrac{2.S}{R^2}$

MODIFICACIONES DEL PROGRAMA (INTEGRAL DEFINIDA)

- Sustituir las líneas 30, 40, 50

```
30 PRINT TAB (15) "                          "
40 PRINT TAB (15) " CALCULO DEL NUMERO PI    "
50 PRINT TAB (15) "                          "
```

- Sustituir las líneas 70, 80 y 90 por

```
70 INPUT  "CUAL ES EL VALOR DEL RADIO";R
80 IF R <= Ø PRINT " ERROR AL INTRODUCIR EL RADIO":GOTO 70
90 A = - R : B = R
```

- Sustituir la línea 170

170 C=X:X=R*R - X * X: GOSUB 3ØØ3Ø:X=C

- Sustituir la línea 21Ø

21Ø IF(A+.ØØØØ1)>=B PRINT "LA INTEGRAL VALE";S:GOTO 23Ø

- Añadir las líneas 230, 240

23Ø PRINT "RADIO", "AREA CIRCULO", "RELACION"

24Ø PRINT R, 2 * S, (2 * S)/(R * R)

- Añadir la subrutina raíz cuadrada líneas de la 3ØØØØ hasta la 3ØØ9Ø.

PROGRAMA

```
10 REM * COCEPTO DE INTEGRAL *
20 CLS
30 PRINT TAB(15);"-------------------------"
40 PRINT TAB(15);"(CALCULO DEL NUMERO PI)"
50 PRINT TAB(15);"-------------------------"
60 PRINT
70 INPUT"CUAL ES EL VALOR DEL RADIO";R
80 IF R<=0 PRINT"ERROR AL INTRODUCIR EL RADIO":GOTO 70
90 A=-R:B=R
100 INPUT"CUAL ES EL INCREMENTO DE LA ABCISA X";H
110 IF H<=0 PRINT"!ERROR! (INCREMENTO NEGATIVO O CERO)":GOTO 100
120 IF(B-A)<H PRINT"DISMINUYA EL INCREMENTO DE X":GOTO 100
130 S=0
132 CLS
135 PRINT"ABCISA","ORDENADA","T. ELEMENTAL","T. TOTAL"
140 IF (B-A+.00001)<H THEN H=B-A
150 X=A+H/2
160 REM * EN LA LINEA SIGUIENTE SE SITUA LA ECUACION *
170 C=X:X=R*R-X*X:GOSUB 30030:X=C
180 S=S+Y*H
190 A=A+H
200 PRINT X,Y,Y*H,S
210 IF (A+.00001)>=B PRINT"LA INTEGRAL VALE";S:GOTO 230
220 GOTO 140
230 PRINT "RADIO","AREA CIRCULO","RELACCION"
240 PRINT R,2*S,(2*S)/(R*R)
```

```
30000 END
30010 REM *RAIZ CUADRADA* INPUT X,OUTPUT Y
30020 REM TAMBIEN USA W & Z INTERNAMENTE
30030 IF X=0 THEN Y=0:RETURN
30040 IF X>0 THEN 30060
30050 PRINT"RAIZ DE UN NUMERO NEGGATIVO":STOP
30060 Y=X*.5:Z=0
30070 W=(X/Y-Y)*.5
30080 IF (W=0)OR(W=Z) THEN RETURN
30090 Y=Y+W:Z=W:GOTO 30070
```

Una vez efectuados los cambios vamos a ejecutar el programa dando distintos valores al radio y a Δx. La tabla que se obtiene es la siguiente :

RADIO	PASO	AREA DEL CIRCULO	A/R^2
1	0,01	3,14194	3,14194
2	0,02	12,5678	3,14194
3	0,03	28,2775	3,14194
5Ø	0,5	7854,84	3,14194

Se ha mantenido la misma relación entre el radio y el paso para evitar diferencias de precisión en el cálculo del área.

La conclusión es inmediata : existe una relación constante entre el área de un círculo y el cuadrado de su radio. Su cociente es constante y el valor de este cociente es lo que conocemos por la constante π .

La precisión en el cálculo de π por el método que acabamos de explicar, es función del valor del paso (Δx) y de la propia precisión del microordenador TRS-80.

De esta forma, para Δx =.001 y R =1, el cálculo de la constante de un valor de 3,14163. El error es menor que 4×10^{-5} (Como se sabe el número π vale 3,1415927...).

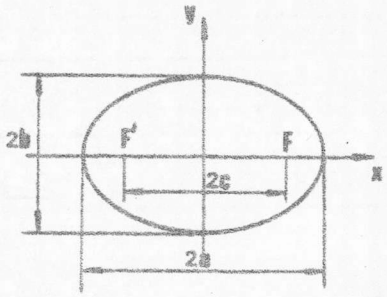
AREA DE LA ELIPSE

Como se sabe, la ecuación de la elipse centrada en sus ejes toma la expresión :

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

ó lo que es lo mismo

$$y = \pm \sqrt{\left(1 - \frac{x^2}{a^2}\right) b}$$



Nuestro objetivo es calcular el área de la elipse.

Vamos a calcularla matemáticamente primero y luego haremos uso de nuestro programa de integral definida para comprobar su perfecto funcionamiento.

$$\text{AREA} = 4 \cdot \int_0^a \sqrt{\left(1 - \frac{x^2}{a^2}\right) b^2} \; dx = 4b \int_0^a \sqrt{\left(1 - \frac{x^2}{a^2}\right)} \; dx$$

utilizando el cambio de variable $\frac{x^2}{a^2} = \text{sen}^2 t$

$dx = a \cos t \, dt$ y modificando los límites de integración

$$A = 4ab \int_0^{\pi/2} \cos^2 t \; dt = 4 \; a \; b \int_0^{\pi/2} \left(\frac{1 + \cos 2t}{2}\right) dt$$

$$A = 2 \; a \; b \int_0^{\pi} (1 + \cos V) \; \frac{dv}{2} = ab \; \Big[V + \text{sen } V\Big]_0^{\pi}$$

$$A = \boxed{\pi \; a \; b}$$

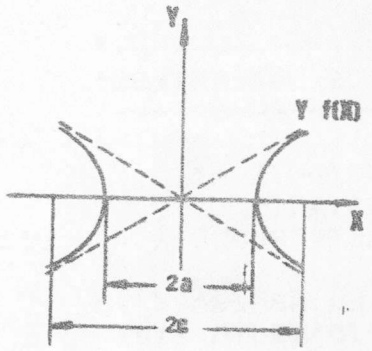
El programa que calcula el área de la elipse, tiene mucha similitud con el programa del cálculo de π.

PROGRAMA

```
10 REM * CALCULO DEL AREA DE UNA ELIPSE *
20 CLS
30 PRINT TAB(15);"--------------------------------"
40 PRINT TAB(15);"CALCULO DEL AREA CE UNA ELIPSE"
50 PRINT TAB(15);"--------------------------------"
60 PRINT
70 INPUT"CUAL ES EL VALOR DE LA CONSTANTE A";A
80 INPUT"CUAL ES EL VALOR DE LA CONSTANTE B";B
90 IF (A<=0)OR(B<=0) PRINT"ERROR AL INTRODUCIR A O B":GOTO 70
100 D=-A:E=A:S=0
110 INPUT"CUAL ES EL INCREMENTO DE LA ABCISA X";H
120 IF H<=0 PRINT"ERROR DE INCREMENTO":GOTO 110
130 IF 2*A<H PRINT"DISMINUYA EL INCREMENTO DE X":GOTO 110
140 CLS
150 PRINT"ABCISA","ORDENADA","T. ELEMENTAL","T. TOTAL"
160 IF (E-D+.00001)<H THEN H=E-D
170 X=D+H/2
175 REM *EN LA LINEA SIGUIENTE SE SITUA LA ECUACION Y=F(X)*
180 C=X:X=B*B*(1-(X*X)/(A*A)):GOSUB 30030:X=C
190 S=S+Y*H
200 D=D+H
210 PRINT X,Y,Y*H,S
220 IF (D+.00001)>=E PRINT"LA INTEGRAL ES";S:GOTO 240
230 GOTO 160
240 PRINT"CONSTANTE A","CONSTANTE B","AREA DE LA PARABOLA"
250 PRINT A,B,S*2
30000 END
30010 REM *RAIZ CUADRADA* INPUT X,OUTPUT Y
30020 REM TAMBIEN USA W & Z INTERNAMENTE
30030 IF X=0 THEN Y=0:RETURN
30040 IF X>0 THEN 30060
30050 PRINT"RAIZ DE UN NUMERO NEGATIVO":STOP
30060 Y=X*.5:Z=0
30070 W=(X/Y-Y)*.5
30080 IF (W=0)OR(W=Z) THEN RETURN
30090 Y=Y+W:Z=W:GOTO 30070
```

Se puede ejecutar el programa para varios valores de A y B. En particular para A = 1 y B = 1 la parábola se convierte en una circunferencia.

De modo parecido, se puede calcular el área de una rama hiperbólica entre dos intervalos. La ecuación de la hipérbola centrada en sus ejes es :



$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$$

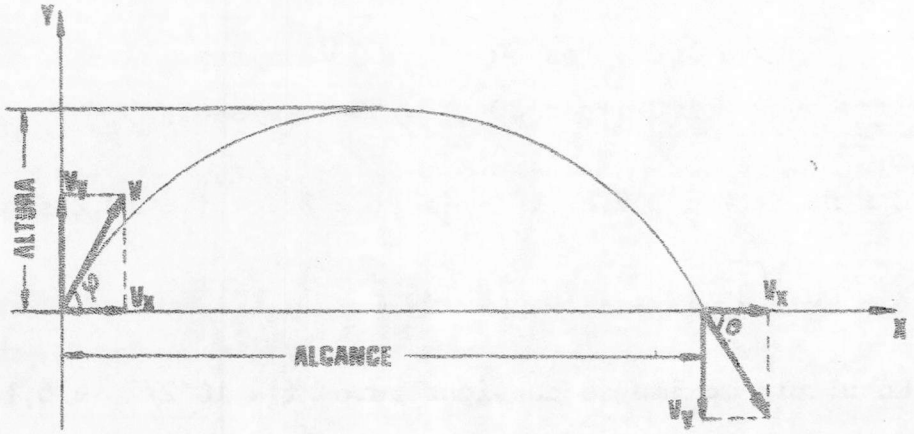
EJERCICIO PROPUESTO

Modificar el programa del cálculo del área de una elipse para calcular el área de una rama hiperbólica dentro de un intervalo.

ESTUDIO DEL TIRO PARABOLICO

El estudio del tiro parabólico parte del concepto del vector velocidad. La velocidad, además de ser una magnitud escalar, puede representarse por un vector cuyo módulo es una magnitud escalar y cuyo argumento viene determinado por su dirección y sentido. Asímismo, la aceleración es una magnitud vectorial puesto que es la derivada de la velocidad con respecto al tiempo.

Supongamos que se lanza un proyectil con una determinada velocidad V y con un ángulo y respecto de la horizontal. La trayectoria constituye un ejemplo típico de tiro parabólico.



Las leyes del movimiento, aplicadas a la descomposición Vx y Vy del vector V, definen con claridad esta trayectoria.

$$\left.\begin{array}{l} Vx = V \cos \varphi \\ Vy = V \operatorname{sen} \varphi \end{array}\right\} \quad \varphi = \operatorname{arc\,tg} \frac{Vy}{Vx}$$

La única fuerza que interviene en el proyectil una vez lanzado es la acción de la gravedad en la dirección vertical. Por tanto, el estudio aislado del desplazamiento vertical corresponde a un movimiento del tipo uniformemente acelerado, mientras que el horizontal es un movimiento uniforme.

Las ecuaciones son las siguientes :

$$\left.\begin{array}{l} y = Vy.t - \frac{1}{2}\ g.t^2 \\ \\ x = Vx.t \end{array}\right\} \quad \begin{array}{l} y = F(t) \\ \\ x = H(t) \end{array}$$

El ángulo de caída del proyectil $\theta = -\text{ arc. tag } \frac{Vy}{Vx}$

Ejemplo :

Se dispara un proyectil con una velocidad inicial de 100 m/s., y un ángulo de elevación de 30º. Se desea calcular el alcance del proyectil, suponiendo un terreno horizontal y la altura máxima y el ángulo de caída.

Aplicando las ecuaciones :

$$Vx = 100\ .\ \cos 30 \quad = 50\sqrt{3}\ \text{m/seg.}$$

$$Vy = 100\ .\ \text{sen } 30 \quad = 50 \quad \text{m/seg.}$$

$$0 = 50.\ t - \frac{1}{2}\ .\ 9{,}8.\ t^2 \longrightarrow t = \emptyset\ , \quad t = 10.2 \text{ seg.}$$

$$x = 50\sqrt{3}\ .\ 10.2 \quad = \quad 883135 \text{ m.}$$

La altura máxima se consigue para $t = 10'2/2 = 5{,}1$ seg.

$$Y\ max = 50\ .\ 5{,}1 - \frac{1}{2}\ 9{,}8\ .\ (5{,}1)^2$$

$$Y\ max = 127{,}55 \text{ m.}$$

El ángulo de caída $\theta = -30º$

Se pretende simular el tiro parabólico, haciendo uso de las funciones gráficas SET y RESET. El programa tiene que dar entrada a los datos del vector velocidad inicial. Se necesita cargar la subrutina coseno y seno si se utiliza un BASIC NIVEL I.
A la vez se desea poder efectuar distintos disparos, modificando el ángulo de disparo (argumento del vector velocidad).

PROGRAMA

```
10 REM*ESTUDIO DEL TIRO PARABOLICO*
20 CLS:T=0
30 PRINT TAB(15);"-----------------------------"
40 PRINT TAB(15);"ESTUDIO DEL TIRO PARABOLICO"
50 PRINT TAB(15);"-----------------------------"
60 PRINT
70 INPUT"CUAL ES EL MODULO DE LA VELOCIDAD M.(0<M<=100)";M
80 IF(M>100)OR(M<=0)THEN PRINT"MODULO FUERA DE LIMITES":GOTO70
90 INPUT"CUAL ES EL VALOR DEL ARGUMENTO A.(0<=A<=90)";A
100 IF(A<0)OR(A>90)THEN PRINT"ARGUMENTO FUERA DE LIMITES":GOTO90
130 IF T<>0 GOTO
140 CLS
160 FOR X=1 TO 59
170 PRINT@900+X,".";
180 NEXT X
190 FOR Y=3 TO 13
200 PRINT@ Y*64+5,".";
210 NEXT Y
220 FOR N=1 TO 6
230 PRINT@128+128*N,100*(6-N);
240 NEXT N
250 FOR N=1 TO 5
255 PRINT@960+N*11,200*N;
260 PRINT@ 965+N*11,":";
270 NEXT N
280 PRINT@ 965,"0";:PRINT@ 193,"M.";:PRINT@ 1021,"M.";
290 FOR H=40 TO 43
291 FOR L=82 TO 85
292 SET(L,H)
293 NEXT L
294 NEXT H
300 X=A:GOSUB 30380:T=M*Y/4.9
310 X=A:GOSUB 30350:B=M*Y
320 X=A:GOSUB 30380:C=M*Y
330 PRINT@1,"TIRO PARABOLICO :";
340 PRINT@19,"MODULO DE V.=";M;
350 PRINT@39,"ARGUMENTO=";A;" GRADOS";
355 X=0:Y=0
360 FOR I=0 TO T STEP .08
370 RESET(X,Y)
380 X=B*I
390 Y=(C*I)-(4.9*I*I)
400 X=INT(X*110/1000+.5)+12
410 Y=42-INT(Y*28/500+.5)
420 SET(X,Y)
430 NEXT I
440 PRINT@79,"                                        ";
450 PRINT@79,"T=";T;:PRINT@94,"X=";T*B;:PRINT@109,"Y=";"0";
460 PRINT@143,"                                                  ";
470 PRINT@143,"CUAL ES EL ANGULO DEL NUEVO DISPARO";
480 INPUT A
490 IF (A<0)OR(A>90) GOTO 460
500 GOTO 300
30000 END
30350 REM *COSENO* INPUT X EN GRADOS,OUTPUT Y
30360 X=X+270:GOSUB 30380:Y=-Y:RETURN
30370 REM*SENO* INPUT X EN GRADOS, OUTPUT Y
30380 IF X>360 THEN X=X/360:X=(X-INT(X))*360
30390 IF X>90 THEN X=X/90:Y=INT(X):X=(X-Y)*90:ON Y GOTO 30410,30420,30430
```

```
30400 X=X/57.29578:GOTO 30440
30410 X=90-X:GOTO 30400
30420 X=-X:GOTO 30400
30430 X=X-90:GOTO 30400
30440 Y=X-X*X*X/6+X*X*X*X*X/120-X*X*X*X*X*X*X/5040
30450 Y=Y+X*X*X*X*X*X*X*X/362880
30460 RETURN
```

Una vez editado el programa, puede ejecutarse con un módulo de velocidad inicial de 100 m/seg., dándo diferentes valores al argumento.

PREGUNTAS

¿ Para qué valor del ángulo se consigue el alcance máximo?

¿ Qué sucede si se disparan con ángulos complementarios, por ejemplo 30º y 60º ?

MODIFICACIONES AL PROGRAMA

Para simular la trayectoria del proyectil se pueden añadir las líneas 355 y 37Ø

```
355 X = Ø: Y = Ø
37Ø RESET (X,Y)
```

Colocando un objetivo en el campo de tiro, se pueden observar las velocidades y ángulos que consiguen batirlo.

Para dibujar el objetivo, añadir al programa las líneas :

```
29Ø FOR H = 4Ø TO 43
291 FOR L = 82 TO 85
292 SET (L,H)
293 NEXT L
294 NEXT H
```

APARTADO 5º

PROGRAMAS DE UTILIDAD.

PRUEBA COMBINADA DE FUNCION Y ROM

El siguiente programa realiza un chequeo de la microcomputadora. Cuando se ejecuta un programa y existe la sospecha de que el fallo está en el mal funcionamiento de la microcomputadora, puede ejecutarse este programa. Primeramente hay que asegurarse de que la microcomputadora esté correctamente alimentada y efectuar la prueba P.M. descrita en el capítulo 26.

El programa se ejecuta en tres fases :

1. Comprobación de la función (requiere aproximadamente 5 segundos)
2. Comprobación de la memoria RAM (requiere unos minutos).
3. Comprobación de la pantalla. Permite verificar el centrado, la distorsión de la línea recta, etc... (si se desea dibujar de nuevo el patrón hay que oprimir la tecla [ENTER]).

Si en cualquier punto la computadora muestra un BREAK AT ## ## ##, se sabrá que algunas de las funciones no se realizan correctamente (fallo de la ROM). En caso de fallo en la RAM, el mensaje BREAK será precedido por "RAM ERROR".

Si no se obtiene un mensaje BREAK (ó iteración infinita), el TRS-80 funciona correctamente.

Este programa puede guardarse en cinta para usos posteriores. El programa es el siguiente :

```
1Ø IN. "ESCRIBA 1, LUEGO OPRIMA ENTER";X
15 CLS:P. AT Ø; "PRUEBA DE FUNCIONAMIENTO DE LA TRS-8Ø"
2Ø READ Y
3Ø DATA 2
4Ø RESTORE
5Ø READ Y
55 F.A. = 1 TO 1ØØØ:N.A
6Ø IF X>YSTOP
7Ø IF X>= YSTOP
8Ø IF Y<XSTOP
9Ø IF Y<= XSTOP
1ØØ F.X=1TO1ØSTEP2
11Ø GOTO 13Ø
12Ø STOP
13Ø GOS.15Ø
14Ø GOTO 16Ø
15Ø RETURN
16Ø ONX GOTO 18Ø
17Ø STOP
18Ø SET(X,Y)
185 IFPOINT(X,Y)G.19Ø
187 STOP
19Ø RESET (X,Y)
2ØØ IFX<>Y-1STOP
21Ø IFY = X + 1G.23Ø
22Ø STOP
23Ø Z=RND(Ø)
24Ø X=1.1:X=INT(X)
245 Y=ABS(X)/2+.5
25Ø IFY=1G.27Ø
26Ø STOP
27Ø REM EVERYTHING IS OK (todo está correcto)
29Ø CLS:P.TAB(5), "TODAS LAS FUNCIONES ESTAN BIEN, LA PRUEBA
    DE RAM ESTA REALIZANDOSE"
3ØØ A= M./4-1:B=Ø
```

```
310 F.Y=1TO8:Q=.5
320 F.B=1TOY:Q=Q *2:N.B
330 F.X=0 TO A:A(X)=Q:N.X
340 F.X=0 TO A:IFA(X)<>QP."RAM ERROR":STOP
350 N.X
360 P.AT68,Q:N.Y:P.AT0; "LA PRUEBA RAM SE HA COMPLETADO"
370 F.A=1TO 2500:N.A
400 CLS:K=1
410 A$=GH
420 F.X=1TO32:P.A$;:N.X
430 F.X=1TO14:P.T.(29):A$:N.X
440 P.AT 469;
450 F.X=1TO9:P.A$;:N.X:P
460 P.T.(21);:F.X=1TO9:P.A$;:N.X
470 P.AT960;
480 F.X=1TO31:P.A$;:N.X
490 IN.B$
500 IFK>0A$=80
510 IFK<0A$=GH
520 K= - K
530 CLS:G.420
```

PROGRAMA EVALUADOR DE EXAMENES

```
 1Ø REM *EVALUADOR DE EXAMENES*
 2Ø CLS
 3Ø A = 1
 4Ø B = 2
 5Ø C = 3
 6Ø D = 4
 7Ø E = 5
 8Ø T = 1Ø
 9Ø F = 11
1ØØ N = 1Ø
11Ø P."LE GUSTARIA ESCRIBIR LAS RESPUESTAS"
12Ø P."UNA CADA VEZ, O 5 CADA VEZ (DIGITE 1 ó 5)";:IN.T
13Ø IF(T=1)+(T=5)GOTO15Ø
14Ø GOTO 2Ø
15Ø UN."INTRODUZCA EL NOMBRE DEL ESTUDIANTE (APELLIDO, NOMBRE)";
    A$;B$
16Ø CLS:P."PRUEBA PARA":B$,"  ";A$
17Ø REST
18Ø R = Ø
19Ø FOR I= 1 TO N STEP T
2ØØ P."INTRODUZCA RESPUESTA";:IF T= 5P."S";I;"A TRAVES";
21Ø P.I+T-1;
22Ø IF T = 5 IN. A(I),A(I+1),A(I+2),A(I+3),A(I+4),A(I+5):G.24Ø
23Ø IN.A(I)
24Ø NEXT I
25Ø CLS
26Ø P."RESULTADOS DE LA PRUEBA DE ";B$;"  ";A$;";"
27Ø FOR I = 1 TO N
28Ø P.A(I)
29Ø READ Z
3ØØ P.Z,
```

```
31Ø IF A(I)=ZP."CORRECTO";:R= R + 1
32Ø P.
33Ø NEXT I
34Ø P. "PORCENTAJE DE CORRECTAS:" ;INT(R/N *1ØØ + .5)
35Ø P.
36Ø G.15Ø
37Ø DATA 5,3,A,D,C,E,T,T,F,T
```

RELOJ DE 12 HORAS

```
 1Ø IN."LA HORA ES";E
 2Ø F=INT(E/1Ø):E=E-(F *1Ø)
 3Ø IN."LOS MINUTOS SON";C
 4Ø D=INT(C/1Ø):C=C-(D * 1Ø)
 5Ø IN. "LOS SEGUNDOS SON";A
 6Ø B=INT(A/1Ø):A=A-(B * 1Ø)
 7Ø F.N=1 TO 3ØØ:N.N
 8Ø A = A + 1
 9Ø IF A>9 G.11Ø
1ØØ G.3ØØ
11Ø A = Ø
12Ø B = B + 1
13Ø IF B>5 G.15Ø
14Ø G.3ØØ
15Ø B = Ø
16Ø C = C + 1
17Ø IF C>9 G.19Ø
18Ø G.3ØØ
19Ø C = Ø
2ØØ D = D + 1
21Ø IF D>5 G.23Ø
22Ø G.3ØØ
23Ø D = Ø
24Ø E = E + 1
25Ø IF E>9 G.27Ø
26Ø G.29Ø
27Ø E = Ø
28Ø F = F + 1
29Ø IF (F=1)* (E=3)A=Ø:B=Ø:C=Ø:D=Ø:E=1:F=Ø
3ØØ CLS
31Ø P.AT 47Ø,F;E;":";D;C;";";B;A
32Ø G.7Ø
```

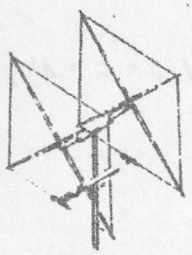
PROGRAMA PARA EL DISEÑO DE UNA ANTENA CUBICA QUAD

La cúbica QUAD es una antena excepcionalmente fina para el uso en la recepción y transmisión de radioaficionados, banda civil, transmisión de onda corta, servicio de radio industrial, público y señales de televisión. Es giratoria y de poco peso hallándose bien equilibrada, aún cuando no se encuentre muy por encima del nivel del suelo.

Eléctricamente, consiste en dos bucles de alambre, uno de los cuales se alimenta con un cable coaxial o uno de doble terminal, y el otro simplemente soldado a sus extremos. La figura 1 es una ilustración de esta antena QUAD.

El programa recibe la frecuencia de operación deseada. Luego calcula todas las dimensiones necesarias para su construcción.

Figura 1



```
 5 CLS
 8 P."DISEÑO A LA MEDIDA DE SU PROPIA ANTENA CUBICA DE ALTA
    GANANCIA>> ->> "
 9 P.
1Ø IN."FRECUENCIA CENTRAL ( EN MEGAHERTZ) = ";F
2Ø CLS
25 P." > -- > ANTENA CUBICA QUAD < -- < "
3Ø P.
4Ø E= .985 * F
5Ø G= 1.Ø33 * F
6Ø D= 1ØØØ/F
1ØØ R= 1Ø32/F
14Ø B= 118/F
18Ø X= (2 *(R *R/64))
19Ø GOSUB 1ØØØØ
195 S = Y
2ØØ X= (S *S + (B* B/4))
21Ø GOSUB 1ØØØØ
215 P = Y
22Ø X = ((R *R/64) + 75 * 75/(F* F *4))
23Ø GOSUB 1ØØØØ
235 T = Y
24Ø X = ((R *R/64) + 125* 125/(F* F *4))
242 GOSUB 1ØØØØ
245 U = Y
26Ø W = 468/F
52Ø P. "LA FRECUENCIA CENTRAL DEL DISEÑO ES";F;"MHZ."
525 P. "ESTA QUAD DE DOS ELEMENTOS DEBERA TENER UNA RAZON DE
    ONDA ESTACIONARIA"
53Ø P. "DE 2:1 O MENOS SOBRE EL RANGO DE FRECUENCIA DE";E;"A"
535 P.G;"MHZ CUANDO SE USE CON UNA LINEA DE ALIMENTACION DE 5Ø
    A 75 OHMS."
54Ø P.
57Ø P. "LA LONGITUD DEL "BOOM" PUEDE VARIAR ENTRE";75/F;"PIES Y"
575 P. 125/F;"PIES CON POCO EFECTO. UNA LONGITUD DE ";B
```

```
58Ø P."PIES ES OPTIMA"
59Ø P.
62Ø P. "LA LONGITUD TOTAL DEL ALAMBRE EN EL ELEMENTO MANEJADO ES";
    D
625 P. "PIES, LO CUAL RESULTA SER"; D/4;"PIES POR CADA LADO"
63Ø P.
64Ø P.T.(2Ø), "OPRIMA ENTER PARA CONTINUAR",:IN;A$
65Ø CLS
66Ø P."LONGITUD TOTAL DEL ALAMBRE EN EL ELEMENTO REFLECTOR ES";R
665 P. "PIES, LO CUAL SERA"; R/4;"PIES POR CADA LADO"
67Ø P.
69Ø P." LA LONGITUD MINIMA DE BAMBU. FIBRA DE VIDRIO O ALGUN OTRO"
695 P. "SERA";S: "PIES, MEDIDOS DESDE EL CENTRO DEL"
697 P. "BOOM'. SI SE USA MONTADURA QUAD DE ARAÑA (SIN BOOM),"
7ØØ P. "CADA BRAZO DEBERA SER POR LO MENOS DE";P;" "PIES"
73Ø P.
74Ø P. "EL RADIO DE GIRO (PARA LIBRAMIENTOS,ETC) VARIARA"
745 P. "ENTRE";T; "PIES Y";U; "PIES, DEPENDIENDO DE LA LONGITUD"
75Ø P. "DEL 'BOOM'"
755 P.
76Ø P.T.(2Ø), "OPRIMA ENTER PARA CONTINUAR";:IN.A$
77Ø CLS
79Ø P. "ESTA ANTENA QUAD TRABAJARA BIEN AUN A BAJAS ALTURAS"
795 P. "POR ENCIMA DEL SUELO, PERO TRABAJA MEJOR CUANDO ESTA
    ELEVADA"
8ØØ P. " A LA MITAD DE LA LONGITUD DE ONDA -- ";W;"PIES,O MAS"
81Ø P.
84Ø P."LA RAZON DEL FRENTE HACIA ATRAS (HABILIDAD PARA REDUCIR
    SEÑALES"
845 P. "NO DESEADAS DE LA DIRECCION OPUESTA) DEBERA EXCEDER
    DE 1Ø"
85Ø P. "DECIBELES DE CERCA DE";.97 *F; "A";1.Ø3* F; "MEGAHERTZ"
855 P. "ACERCANDOSE A 25 DB A"; F;"MEGAHERTZ"
86Ø P.
88Ø P. "* * * * *MUYBIEN* * * * *"
```

```
 9999 END
1ØØØØ REM * SUBRUTINA DE RAIZ CUADRADA*
1ØØ1Ø IF X  = Ø GOTO 1ØØ4Ø
1ØØ2Ø P. "NO EXISTE TAL COSA COMO RAIZ CUADRADA DE NUMERO NEGATIVO"
1ØØ3Ø END
1ØØ4Ø Y = X/Z
1ØØ5Ø Z = Ø
1ØØ6Ø W = (X/Y - Y)/2
1ØØ7Ø IF W = Ø RET.
1ØØ8Ø IF W = Z RET.
1ØØ9Ø Y = +W:Z=W
1Ø1ØØ GOTO 1ØØ6Ø
```

PROGRAMA PARA MEDIR LA VELOCIDAD DE LECTURA

```
3 REM * PROGRAMA DE LECTURA VELOZ *
4 G. 1Ø
5 F. I =1 TO B:N.I:PRINT AT 448;:RET.
6 REM *EL APUNTADOR AUDIBLE VA ANTES DEL REGRESO EN LA LINEA
  ANTERIOR *
1Ø I. "CUANTAS PALABRAS POR MINUTO LEE?";W
2Ø B = (12 * 6Ø/W) * 5ØØ
3Ø REM 5ØØ = FOR/NEXT ITERACIONES EN UN SEGUNDO
4Ø CLS
5Ø P.AT 448;
1ØØ P. "SCARLETT O'HARA NO ERA BONITA, PERO LOS HOMBRES RARA VEZ":
  GOS.5
1Ø2 P. "LO NOTABAN CUANDO ERAN ATRAIDOS POR SU PROPIA SIMPATIA
  COMO LES PASO": GOS.5
1Ø4 P. "A LOS MELLIZOS TARLETON. EN SU CARA ESTABAN DEMASIADO
  MEZCLADOS LOS ":GOS.5
1Ø6 P. "RASGOS DELICADOS DE SU MADRE, UNA ARISTOCRATA COSTEÑA DE":
  GOS.5
1Ø8 P." DESCENDENCIA FRANCESA, Y LOS TOSCOS DE SU ENROJECIDO PADRE"
  :GOS.5
11Ø P. "IRLANDES. PERO ERA UNA CARA IMPRESIONANTE, SU MENTON
  PRONUNCIADO,":GOS.5
112 P. "SU QUIJADA RECTANGULAR. SUS OJOS ERAN DE UN VERDE PALIDO
  SIN UN TOQUE":GOS.5
114 P. "COLOR AVELLANA. CON NEGRAS PESTAÑAS ERIZADAS MUY IMPRE-
  SIONANTES Y LIGERAMENTE":GOS.5
116 P. "MARCADAS EN LOS EXTREMOS. SOBRE ELLAS, SUS CEJAS ESPESAS
  Y NEGRAS":GOS.5
118 P. "SESGADAS HACIA ARRIBA, CORTANDO UNA DESTELLANTE LINEA
  OBLICUA EN SU":GOS.5
12Ø P. "PIEL BLANCO MAGNOLIA - ESA PIEL TAN APRECIADA POR LAS
  MUJERES":GOS.5
122 P. "DEL SUR Y TAN CUIDADOSAMENTE GUARDADA CON VELOS, COFIAS
  Y":GOS.5
124 P. "MITONES EN CONTRA DEL CALIENTE SOL DE GEORGIA":GOS.5
```

PROGRAMA PROMEDIO DE DOW-JONES

Es un ejemplo aparecido en una revista financiera en 1977 :

```
 1Ø REM *DE FORBES 6/1/77,P.9Ø. ARTICULO DE BROWN *
 2Ø P. "*** PROYECTOS CON BLANCO EN EL PROMEDIO INDUSTRIAL
    DOW-JONES COMO A"
 3Ø P. "FUNCION DE LAS GANANCIAS DJI Y TASA DE INFLACION  *** "
 4Ø P.
 5Ø REM *KILO= PRIMARDECOSTO DEL DINERO. SUPONGA 3% *
 6Ø K = .Ø3
 7Ø REM *P= PRIMA DEL RIESGO DE ACCIONES SOBRE BONOS.SUPONGA 1% *
 8Ø P = .Ø1
 85 Y = 1
 86 N = Ø
 9Ø P. "CONOCE LAS GANANCIAS ESPERADAS ANUALES DE 3Ø DJI (Y/N)";
1ØØ INPUT A
11Ø IF A = 1 THEN 27Ø
12Ø P.
13Ø P. "ESTE METODO LE DARA UNA APROXIMACION DE LAS GANANCIAS
    UTILIZANDO LOS "
14Ø P. "PRECIOS DEL PERIODICO Y LAS TASAS P/E. MEJORES PREDIC-
    CIONES"
145 P. "DE CADA UNA DE LAS GANANCIAS DE LAS COMPAÑIAS PODRAN
    DAR UN MEJOR"
15Ø P. "PRONOSTICO GENERAL"
16Ø P.
17Ø D = Ø
175 FOR N = 1 TO 3Ø
18Ø READ A $
2ØØ P. "CUAL ES EL PRECIO ACTUAL DE >-->". A$," <--<";
21Ø INPUT P
22Ø P. "LA TASA P/E ACTUAL":
23Ø INPUT R
24Ø E = P/R
```

```
25Ø D* E + D
26Ø N.N.
265 P.
266 G.31Ø
27Ø P. "CUALES SON LAS GANANCIAS TOTALES ESPERADAS PARA UNA
    ACCION DE";
28Ø P. "CADA UNA";
285 INPUT D
29Ø REM *I= TASA DE INFLACION ESTIMADA*
31Ø P. "QUE PORCENTAJE ES LA TASA DE INFLACION?";
32Ø INPUT I
33Ø T = D/(K + P + I *.Ø1)
34Ø R + T/D
35Ø P.
36Ø P. "TASA INFL.", "GAN.DJI", "PROM DJ ESTIM", "TASA PROM/GAN"
37Ø P.
38Ø P. I,D,T,R
39Ø D. ALLIED, CHEM, ALCOA, AMER BRANDS, AMER CAN, A.T. & T
4ØØ D. BETH STEEL, CHRYSLER, DUPONT, E. KODAK, ESMARK, EXXON
41Ø D. G.E., GEN FOODS, GEN MOTORS, GOODYEAR, INCO
42Ø D. INT.HARV., INT.PAPER, JOHNS-MAN, MINN MM, OWENS-ILLS
43Ø D. PROCTER & G., SEARS, STD OIL CAL, TEXACO, UNION CARBIDE
44Ø D. U.S STEEL, UNITED TECHNOL., WESTINGHOUSE, WOOLWORTH
```

PROGRAMA DE ENTRETENIMIENTO

```
  4Ø CLS
  5Ø P.AT7, "EN UNA TARDE NEVADA... POR ROBERT FROST":
  55 F.N =1 TO2ØØØ:N.N
  68 F. Z= 1 TO 3ØØ
  7Ø SET (RND(127),RND(47))
  72 N.Z
  8Ø I = Ø
1ØØØ P.AT525, "DE QUIEN SON ESTOS BOSQUES CREO QUE LO SE";
1ØØ1 GOSUB 6ØØØ
11ØØ P.AT525, "SIN EMBARGO, SU CASA ESTA EN LA VILLA";
11Ø1 GOSUB 6ØØØ
12ØØ P. AT525, "EL NO ME VERA DETENERME AQUI";
12Ø1 GOSUB 6ØØØ
13ØØ P. AT525, "PARA OBSERVAR SUS BOSQUES AL CUBRIRSE DE NIEVE";
13Ø1 GOSUB 6ØØØ
14ØØ P. AT525, "MI PEQUEÑO CABALLO HA DE CREER QUE ES EXTRAÑO";
14Ø1 GOSUB 6ØØØ
15ØØ P. AT525, "DETENERSE SIN ALGUNA GRANJA CERCA";
15Ø1 GOSUB 6ØØØ
16ØØ P. AT525,"ENTRE EL BOSQUE Y EL LAGO CONGELADO" ;
16Ø1 GOSUB 6ØØØ
17ØØ P. AT525,"LA TARDE MAS OSCURA DEL AÑO";
17Ø1 GOSUB 6ØØØ
18ØØ P. AT525, "LE DA A LAS CAMPANAS DE SU RIENDA UN JALON";
18Ø1 GOSUB 6ØØØ
19ØØ P. AT524, "PARA PREGUNTAR SI EXISTE ALGUN ERROR";
19Ø1 GOSUB 6ØØØ
2ØØØ P. AT525, "EL UNICO SONIDO ADEMAS DE ESTE ES EL ROCE";
2ØØ1 GOSUB 6ØØØ
21ØØ P. AT525, "DEL VIENTO SUAVE Y DE LA ESCARCHA BLANDA";
21Ø1 GOSUB 6ØØØ
```

```
22ØØ P. AT525, "EL BOSQUE ES ADORABLE, OSCURO Y PROFUNDO";
22Ø1 GOSUB 6ØØØ
23ØØ P. AT589, "PERO TENGO PROMESAS QUE GUARDAR";
23Ø5 I = 3
231Ø GOSUB 6ØØØ
24ØØ P. AT635, "Y MILLAS QUE RECORRER ANTES DE QUE YO DUERMA";
24Ø5 I = 6
241Ø GOSUB 6ØØØ
25ØØ P. AT717, "Y MILLAS QUE RECORRER ANTES DE QUE YO DUERMA";
25Ø5 I = 9
251Ø GOSUB 6ØØØ
5ØØØ SET(RND(127),RND(47))
5ØØ1 G.5ØØØ
6ØØØ F. N = 1 TO 2Ø
6Ø2Ø X =RND(127)
6Ø3Ø Y =RND(47)
6Ø7Ø IF Y = 24 + IG. 6Ø2Ø
6Ø8Ø IF Y = 25 + I G.6Ø2Ø
6Ø9Ø IF Y = 26 + I G.6Ø2Ø
61ØØ SET(X,Y)
615Ø F. A = 1 TO 2Ø:N.A
62ØØ N.N
63ØØ RETURN
```

SUBRUTINAS CONTENIDAS EN ESTE APARTADO

Raíz cuadrada
Exponenciación
Logaritmos (Natural y Común)
Exponenciación (Potencias de e)
Tangente
Coseno
Seno
Arco-Seno
Arco-Coseno
Arco-Tangente
Signo

RAIZ CUADRADA

Calcula : SQR(X), $\sqrt{X}$

Entrada: X, deberá ser mayor o igual a cero.

Salida: Y

Tambien utiliza: internamente W,Z

Otras subrutinas requeridas: ninguna

Como llamarla?: GOSUB 30030

```
30000 END
30010 REM * RAIZ CUADRADA * INPUT X,OUTPUT Y
30020 REM TAMBIEN USA W & Z INTERNAMENTE
30030 IF X=0 THEN Y=0:RETURN
30040 IF X>0 THEN 30060
30050 PRINT"RAIZ DE UN NUMERO NEGATIVO":STOP
30060 Y=X*.5:Z=0
30070 W=(X/Y-Y)*.5
30080 IF(W=0)OR(W=Z) THEN RETURN
30090 Y=Y+W:Z=W:GOTO 30070
```

Exponenciacion

Calcula: X^Y (X a la potencia Y)

Entrada: X,Y. Si X es menor que 0, Y deberá ser un entero non.

Salida: P

Tambien utiliza: E,L,A,B,C, internamente

Otras subrutinas requeridas: Log y Exponencial

Como llamarla: 30120

```
30100 REM*EXPONENCIACION* INPUT X,Y:OUTPUT P
30110 REM*TAMBIEN USA INTERNAMENTE E,L,A,B,C*
30120 IF X=0 THEN P=0:RETURN
30130 P=1:IF Y=0 THEN RETURN
30140 IF(X<0)AND(Y-INT(Y)=0) THEN P=1-2*Y+4*INT(Y/2):X=-
30150 GOSUB 30170:X=Y*L:GOSUB 30250
30160 P=P*E:RETURN
```

ritmos (natural y común)

lcula: LOG(X) base e,y LOG (X) base 10

ada: X mayor que ,o igual a cero

a lu: L es el logaritmo natural (base e), X es el logaritmo común (base 10)

ambien utiliza: A,B,C, internamente

s subrutinas requeridas: Ninguna

o llamarla: GOSUB 30190

```
30170 REM*NATURAL & COMUN *INPUT X,OUTPUTL,X
30175 REM SALIDA L ES LOG NATURAL, SALIDA X ES LOG COMUN
30180 REM TAMBIEN UTILIZA A,B,C INTERNAMENTE
30190 E=0:IF X<0 THEN PRINT'LOG INDEFINIDO EN';X;STOP
30195 A=1:B=2:C=.5
30200 IF X>=A THEN X=C*X:E=E+A:GOTO 30200
30205 IF X<C THEN X=B*X:E=E-A:GOTO 30205
30210 X=(X-.707107)/(X+.707107):L=X*X
30215 L=(((.398979*L+.961471)*L+2.88539)*X+E-.5)*.693147
30220 IF ABS(L)<1E-6 THEN L=0
30225 X=L*.4342945:RETURN
```

ponencial

a ua: ESP(X) (e a la potencia X)

ada: X

alida: E

bien utiliza: L,A internamente

tras subrutinas requeridas: Ninguna

o llamarla: GOSUB 30250

```
30240 REM*EXPONENCIAL* INPUT X,OUTPUT E
30245 REM TAMBIEN USA INTERNAMENTE A,B *
30250 IF X=0 THEN E=1:RETURN
30255 IF INT(1.4427*X)>126 THEN PRINT 'SOBRERRANGO':STOP
30260 IF X<0 GOTO 30290
30265 E=1:A=1:B=0
30270 IF A<.00001 THEN RETURN
30275 A=A*X/(B+1):E=E+A:B=B+1:GOTO 30270
30280 X=-X:GOSUB 30265:E=1/E:RETURN
```

gente

alcula: TAN (X)

rada: X en grados mayor o igual a cero

lida: Y

ambién utiliza: A,C internamente

as subrutinas necesarias: Coseno y Seno

Como llamarla: GOSUB 30320

```
30000 END
30300 REM * TANGENTE *INPUTX EN GRADOS,OUTPUT Y
30310 REM TAMBIEN USA A,C INTERNAMENTE
30320 A=X:GOSUB 30360
30330 IF ABS(Y)<1E-5 THEN PRINT'TANGENTE INDEFINIDA':STOP
30340 C=Y:X=A:GOSUB 30380:Y=Y/C:RETURN
```

oseno

cula: COS(X)

rada: X en grados mayor = a cero

lida: Y

bién utiliza: ninguna otra variable

Otras subrutinas necesarias: Seno

no llamarla: GOSUB 30360

```
30350 REM * COSENO *INPUT X,OUTPUT Y
30360 X=X+270:GOSUB 30380:Y=-Y:RETURN
```

Seno

Calcula: SIN (X)

Entrada: X en grados mayor o igual a cero

Salida: Y

También utiliza: ninguna otra variable

Otras subrutinas requeridas: ninguna

Como llamarla: GOSUB 30380

```
30370 REM * SENO *INPUT X,OUTPUT Y
30380 IF X>360 THEN X=X/360:X=(X-INT(X))*360
30390 IF X>90 THEN X=X/90:Y=INT(X):X=(X-Y)*90:ON Y GOTO 30410,30420,30430
30400 X=X/57.29578:GOTO 30440
30410 X=90-X:GOTO 30400
30420 X=-X:GOTO 30400
30430 X=X-90:GOTO 30400
30440 Y=X-X*X*X/6+X*X*X*X*X/120-X*X*X*X*X*X*X/5040
30445 RETURN
30450 Y=Y+X*X*X*X*X*X*X*X*X/362880
```

Arco Coseno

Calcula: arco coseno(S), ángulo cuyo coseno es S

Entrada: S,0 <=S<= 1

Salida: Y en grados, W está en radianes

También utiliza: X, Z internamente

Otras subrutinas requeridas: arcoseno

Como llamarla: GOSUB 30500

```
30000 END
30500 REM * ARCO COSENO *INPUT S,OUTPUT Y,W
30510 REM Y ESTA EN GRADOS,W EN RADIANES
30520 GOSUB 30550:Y=90-Y:W=1.570796-W:RETURN
```

Arco-Seno

Calcula: Arco-seno(S), angulo cuyo seno es S

Entrada: S,0 = S #1

Salida: Y en grados, W en radianes

Tambien utiliza: X,Y internamente

Otras subrutinas requeridas: ninguna

Como llamarla: 30550

```
30530 REM * ARCO SENO *INPUT S,OUTPUT Y,W
30535 REM Y ESTA EN GRADOS ,W ESTA EN RADIANES
30540 REM TAMBIEN USA VARIABLES X,Z INTERNAMENTE
30550 X=S:IF ABS(S)<=.707107 THEN 30610
30560 X=1-S*S:IF X<0 THEN PRINT S;'ESTA FUERA DE RANGO':STOP
30570 W=X/2:Z=0
30590 Y=(X/W-W)/2:IF (Y=0)+(Y=Z) THEN X=W:GOTO 30610
30600 W=W+Y:Z=Y:GOTO 30580
30610 Y=X+X*X*X/6+X*X*X*X*X*X*.075+Y*X*X*X*X*X*X*X*X*4.464286E-2
30620 W=Y+X*X*X*X*X*X*X*X*X*X*X*3.038194E-2
30625 IF ABS(S)>.707107 THEN W=1.570796-W
30630 Y=W*57.29578:RETURN
```

Arco Tangente

Calcula: ATN

Entrada: X

Salida: C en grados, A en radianes

También utiliza: B,T internamente

Otras subrutinas requeridas: Signo

Como llamarla: GOSUB 30690

```
30660 REM * ARCO TANGENTE *INPUT X,OUTPUT C,A
30670 REM C ESTA EN GRADOS ,A ESTA EN RADIANES
30680 REM TAMBIEN USA B,T INTERNAMENTE
30690 GOSUB 30810:X=ABS(X):C=0
30700 IF X=1 THEN C=1:X=1/X
30710 A=X*X
30720 B=((2.36623E-3*A-1.61657E-2)*A+4.29096E-2)*A
30730 B=((((B-7.5289E-2)*A+.106563)*A-.142089)*A+.199936)*A
30740 A=((B-.333332)*A+1)*X
30750 IF C=1 THEN A=1.570796-A
30760 A=T*A:C=A*57.29578:RETURN
```

S no

Calcula: SGN(X), la componente del signo de X

E ra la: X

S lida: T igual a menos uno para X negativa, 0 para X cero, más uno(+1) para X positiva.

Tambien utiliza: Ninguna otra variable

C as subrutinas requeridas: Ninguna

Como llamarla: GOSUB 30810

```
30800 REM * SIGNO *INPUT X,OUTPUT T=-1,0,+1
30810 IF X<0 THEN T=-1
30820 IF X=0 THEN T=0
30830 IF X>0 THEN T=1
30840 RETURN
```

Conexiones para las Terminales de las Tarjetas Laterales de la Abertura de Expansión (Expansión - Port Edge Card)

P/N	NOMBRE DE LA SEÑAL	DESCRIPCION
1	RAS*	salida del estroboscopio de direcciones de las filas para RAMS dinámicas de 16 terminales
2	SYSRES *	salida del reajuste del sistema, bajo durante el inicio del encendido o con el reajuste oprimido.
3	CAS *	salida del estroboscopio de direcciones de las columnas para RAMS de 16 terminales
4	A10	salida de la dirección
5	A12	salida de la dirección
6	A13	salida de la dirección
7	A15	salida de la dirección
8	GND	Tierra de la señal
9	A11	salida de la dirección
10	A14	salida de la dirección
11	A8	salida de la dirección
12	OUT *	salida del estroboscopio periférico de escritura
13	WR*	salida del estroboscopio de escritura de memoria
14	INTAK*	salida del reconocimiento interrumpido
15	RD *	salida del estroboscopio de lectura de la memoria
16	MUX	salida de control del multiplexor para RAMS de 16 terminales (dinámicas)
17	A9	salida de la dirección
18	D4	Bus de datos bidireccional
19	IN*	salida del estroboscopio periférico de lectura
20	D7	Bus de datos bidireccional
21	INT*	entrada del interruptor (enmascarillable)
22	D1	Bus de datos bidireccional
23	TEST *	un "0" lógico en las entradas de tres estados del TEST en A0-A15, D0-D7, WR, RD, IN, OUT, RAS, CAS, -MUX.
24	D6	Bus de datos bidireccional
25	A0	salida de la dirección
26	D3	Bus de datos bidireccional
27	A1	salida de la dirección
28	D5	Bus de datos bidireccional
29	GND	tierra de la señal
30	D0	Bus de datos bidireccional
31	A4	Bus de dirección
32	D2	Bus de datos bidireccional
33	WAIT*	entrada de espera del procesador, para permitir memoria lenta.
34	A3	salida de la dirección
35	A5	salida de la dirección
36	A7	salida de la dirección
37	GND	tierra de la señal
38	A6	salida de la dirección
39	+5V	salida de 5 volt (corriente limitada)
40	A2	salida de la dirección

NOTA: * Significa negativo ("0" lógico) entrada verdadera o salida

Terminales del conector de la tarjeta lateral AMP P/N 88103 -1

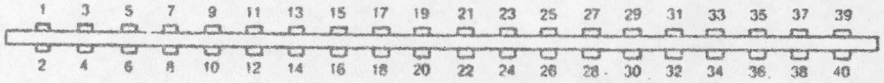


Figura 2. Puntos de conexión para la tarjeta lateral de la abertura de expansión (vista desde la parte trasera del teclado)

Epecificaciones de las Interfaces

Casete

Nivel de entrada sugerido para la reproducción de la grabadora	2 V pico-a-pico a una impedancia Mínima de 360K ohms.
Salida típica de la computadora Nivel a la grabadora	800 m V pico-a-pico a 1K ohm
Capacidad de conmutación remota apagado/encendido	0.5 A max a 6 VDC

Conexiones de las terminales del conector DIN(ver la figura 1)

1— Remoto
2— Tierra de la Señal
3— Remoto
4— Entrada del Borne del Audifono de la Grabadora
5— Salida al Borne Aux o Mic de la Grabadora

Señal de Video

Conexiones de las terminales del borne DIN (ver la figura 1)

1— +5 VDC a 50 mA
2— No utilizada
3— No utilizada
4— Señal de video 1.4V pico a pico, 0.4 V sync negativo, 75 ohms
5— Tierra

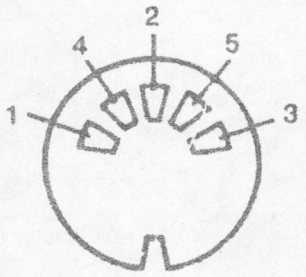


Figura 1. Conexiones de las terminales para los bornes de TAPE (cinta) y VIDEO (DIN) (vistos de la parte trasera del teclado)